AFRL-RI-RS-TR-2017-183

JOINT PROBABILISTIC REASONING ABOUT COREFERENCE AND RELATIONS OF UNIVERSAL SCHEMA

UNIVERSITY OF MASSACHUSETTS AMHERST

OCTOBER 2017

FINAL TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

STINFO COPY

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE

■ **AIR FORCE MATERIEL COMMAND** ■ **UNITED STATES AIR FORCE** ■ **ROME, NY 13441**

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09. This report is available to the general public, including foreign nationals. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RI-RS-TR-2017-183 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE CHIEF ENGINEER:

/ S /
WALTER V. GADZ, JR.
Work Unit Manager

/ S /
MICHAEL J. WESSING
Deputy Chief, Information Intelligence
Systems and Analysis Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
OCTOBER 2017	FINAL TECHNICAL REPORT	OCT 2012 – MAY 2017

4. TITLE AND SUBTITLE

JOINT PROBABILISTIC REASONING ABOUT COREFERENCE AND RELATIONS OF UNIVERSAL SCHEMA

5a. CONTRACT NUMBER

5b. GRANT NUMBER
FA8750-13-2-0020

5c. PROGRAM ELEMENT NUMBER
62303E

6. AUTHOR(S)

Andrew McCallum, Nicholas Monath

5d. PROJECT NUMBER
DEFT

5e. TASK NUMBER
12

5f. WORK UNIT NUMBER
13

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
University of Massachusetts Amherst
Office of Grant and Contract Administration
Venture Way Center, 100 Venture Way, Suite 201
Hadley, MA 01035-9450

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Air Force Research Laboratory/RIEA
525 Brooks Road
Rome NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL/RI

11. SPONSOR/MONITOR'S REPORT NUMBER
AFRL-RI-RS-TR-2017-183

12. DISTRIBUTION AVAILABILITY STATEMENT
Approved for Public Release; Distribution Unlimited. This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

In this project, McCallum's IESL lab at UMass Amherst researched and developed technologies for (1) automatic construction of knowledge bases from natural language text corpora, as well as (2) inference on these knowledge bases. Our work proposes and advances Universal Schema, which jointly learns embedded vector representations for the union of all input schema types (relation types, entity types, and entities themselves), including those from existing knowledge bases (such as Freebase and Wikipedia) as well as relations and types in natural language textual patterns. We present techniques for relation and type prediction based on matrix factorization.

15. SUBJECT TERMS

Relation Extraction, Universal Schema, knowledge base, inference

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			**WALTER V. GADZ, JR.**
U	U	U	UU	78	19b. TELEPHONE NUMBER (*Include area code*) **N/A**

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Table **Page**

1.0 SUMMARY

In this project, McCallum's IESL lab at UMass Amherst researched and developed technologies for (1) automatic construction of knowledge bases from natural language text corpora, as well as (2) inference on these knowledge bases. Our work proposes and advances Universal Schema, which jointly learns embedded vector representations for the union of all input schema types (relation types, entity types, and entities themselves), including those from existing knowledge bases (such as Freebase and Wikipedia) as well as relations and types in natural language textual patterns. We present techniques for relation and type prediction based on matrix factorization. We extended these techniques using neural network models to generalize to even entity, relation and type, unseen at training time. We present a full pipeline of efficient, scalable and accurate algorithms for extracting entity mentions, resolving entities, discovering relations between entities, predicting entity types, learning representations, and performing inference on complex structured knowledge bases.

Highlights of this work include:

- This single PI project had 17 publications in top-tier conferences, or 33 in conferences and peer reviewed workshops combined. Five of these publications have over 50 citations as of August 2017, including the Universal Schema paper, which has over 200 citations.
- This work also significantly advanced development of the *FACTORIE* machine learning and NLP toolkit, which is now widely used; for example it forms part of multiple shipping products from Oracle, Inc.
- Development of Universal Schema based approaches for relation extraction with extensions for generalizing to unseen entities and relations that require only limited amounts of labeled training data. These methods were instrumental in UMass ranking 1st in Spanish TAC-KBP in 2016, a setting with limited resources.
- Methodological advances in named entity recognition, within document coreference, word representations, and relation prediction that defined new state of the art metrics on standard datasets.

2.0 INTRODUCTION

Automatic knowledge base construction (AKBC) is the task of building or extending an existing knowledge base (KB) by means of information extraction from natural language documents and other raw data sources. KBs typically consist of a schema with three classes of objects: entities (such as *Barack Obama, Dell Inc., and Lincoln, Nebraska*), entity types (such as *president, author, privately-held-company, county-seat-in-usa*), and relations between entities (such as *president-of, married-to, headquartered-in, resident-of*).

A limiting factor of many AKBC systems is the use of a rigid and fixed schema of relations and types. Defining a fixed schema in such a way that contains *all* relationships between entities that would ever be desired is quite challenging. For instance, Freebase [1], despite its wide coverage, has no *criticized* or *scientist-at* relations. Of course, a fixed schema also does not provide flexibility for adding new relations or support for the use of arbitrary natural language textual patterns as

relations (e.g. …*adamantly criticized*… or … *considered for tenure at* …). In our work, we present a knowledge base construction system developed around *Universal Schema* [2][3][4][5], in which we use the union of all available knowledge base schemas as well as textual pattern strings to define an ever growing schema for relation and entity types.

We develop a series of techniques based on probabilistic models of matrix factorization and collaborative filtering to predict relationships between pairs of entities and to predict entity types for entities. These models learn low dimensional representations of entities, entity pairs, types, and relations such that mathematical operations in the learned embedding space correspond to semantics of the objects involved and can be used for prediction, inference, and logic of discovering/adding facts of a knowledge base.

Unlike many relation extraction systems, the Universal Schema based approaches can benefit from a large amount of unlabeled data. Our approaches do not require manual annotation in cases where other systems do, as demonstrated in our multilingual experiments in the Results and Discussion section. We describe extensions of the initial matrix factorization models, which provide generalization to make predictions about *any* textual relation or entity type (even those unseen at training time) with recurrent neural networks [4]. We further extend this work to allow for the generalization to unseen entities and entity pairs by means of a neural network attention model over textual evidence in the knowledge base [5].

To construct knowledge bases with Universal Schema, we use a three-stage approach. We ingest a large amount of natural language documents (such as newswire, web-forums, etc). We first apply entity extraction techniques to determine the tokens in each document corresponding to an entity mention. These techniques include log linear model and conditional random field based approaches [6] [7] as well as neural network based approaches [8]. Next, we apply entity resolution techniques based on embedding space representations [9] to link these possibly ambiguous entity mentions to entities in Wikipedia or Freebase. Lastly, after identifying pairs of co-occurring entities with text spans between them, we can used trained Universal Schema models to make predictions about the entities held between the entity pairs [4] [9] [10].

Universal Schema and similar approaches rely on low dimensional *embedded* or *distributed* vector representations of objects such as words, entities, and relations. We present work for representing multiple senses of words by storing a non-parametric number of different vectors for each word type[11]. We also present work that learns *Gaussian* embeddings, which can naturally capture asymmetric similarities, entailment, and uncertainty in representations [12].

Universal Schema presents a relation prediction mechanism that only considers a single relation / entity type “edge” in the vast knowledge graph at a time. However, we would like to be able to make use of multiple edges or a path between entities in the knowledge graph. Using embedded low-dimensional representations of the elements of the knowledge graph, we present neural network based approaches that serve as a generalized version of Horn-clause like logical inference. We present recurrent neural network models for encoding chains of reasoning for prediction/inference in knowledge bases [13] [14]. Finally, we extended the Neural Programmer model [15], which induces latent programs, to perform question answering on knowledge base fact tables [16].

3.0 METHODS, ASSUMPTIONS, AND PROCEDURES

3.1. AKBC and Relation Extraction

3.1.1 AKBC and Relation Extraction Overview

AKBC extracts unary attributes of the form *(subject, attribute)*, typed binary relations of the form *(subject, relation, object)*, or higher-order relations. We refer to subjects and objects as entities. We examine unary prediction for entity types (such as *CEO* or *architect*) and binary relations (such as *married-to* and *headquartered-in*). In many cases, higher order relations can be expressed as collections of binary relations as in Freebase [1].

The task of predicting relationships between pairs of entities, *relation extraction*, is often thought of as a link prediction problem in a network. A knowledge base is naturally described as a network, in which entities are nodes and relations are labeled edges [1][17]. Another common setup for relation extraction is sentence classification. In this case, given an individual sentence, and pre-specified entities, a classifier predicts whether the sentence expresses a relation from a target schema. Often these two approaches are used with *fixed* schema relation extraction, we are also interested in predicting new relations or arbitrary natural language textual patterns as relations. Like Universal Schema, some work on this open-domain information extraction (OpenIE) treats any textual pattern as a relation [18] [19][20]. However, it can be difficult to use OpenIE information downstream in tasks that expect a fixed schema.

In our work on Universal Schema, we unify these three methodologies. We use network information and link prediction techniques to infer relations between entities. However, we use a network that includes OpenIE/natural language textual patterns. We also have methods that allow the learned Universal Schema representations to act as sentence classifiers.

3.1.2 Universal Schema

Universal Schema based approaches for relation extraction and entity type prediction [2][3][4][5] jointly embed the union of all available schema types—not only types from multiple structured databases (such as Freebase or Wikipedia info-boxes), but also types expressed as textual patterns from raw text. Universal Schema uses a probabilistic matrix factorization / collaborative filtering based model. In the relation extraction case, we consider a matrix in which rows correspond to entity pairs and columns correspond to relations. Similarly, in the entity type case, rows correspond to entities and columns correspond to entity types. As both cases use the same underlying mathematical model, we illustrate the model in terms of relation extraction here and present the type model briefly at the end of this section.

One reason this approach is particularly powerful is that *r* may be a relation from *any* schema or a textual pattern (… *was married to*…). Since the observed relationships between entities in a manually constructed knowledge base will be sparse and incomplete, using textual patterns in this collaborative filtering style approach is intuitively motivated. As an example, consider the entity

pair (John F. Kennedy, Robert F. Kennedy) and the pair (Krzysztof Choromanski, Anna Choromanska), the source knowledge base might contain the fact that Kennedy's exhibit the relationship *per:sibling-of*, but this might be missing for the lesser known pair of computer scientists. However, we might observe similar textual patterns between these entity pairs (such as *…sibling…*, *…brother of...*), which would naturally have representations similar to *per:sibling-of*, and hence we might predict the pair exhibit this relation. See Figure 1 for more on Universal Schema.

We now describe the probabilistic matrix factorization approaches used. Given a relation *r* between subject entity *s* and object entity *o*, the latent factor model Universal Schema approach is defined by:

$$P(y_{(s,r,o)} = 1) = \sigma(u_{s,o}^T v_r) \qquad \textbf{(1)}$$

where $y_{(s,r,o)} = 1$ indicates that entities *s* and *o* have relation *r*, $\sigma(\cdot)$ is the sigmoid function, $u_{s,o}$ is an embedding for the entity pair *(s,o)* and v_r is an embedding for the relation *r*.

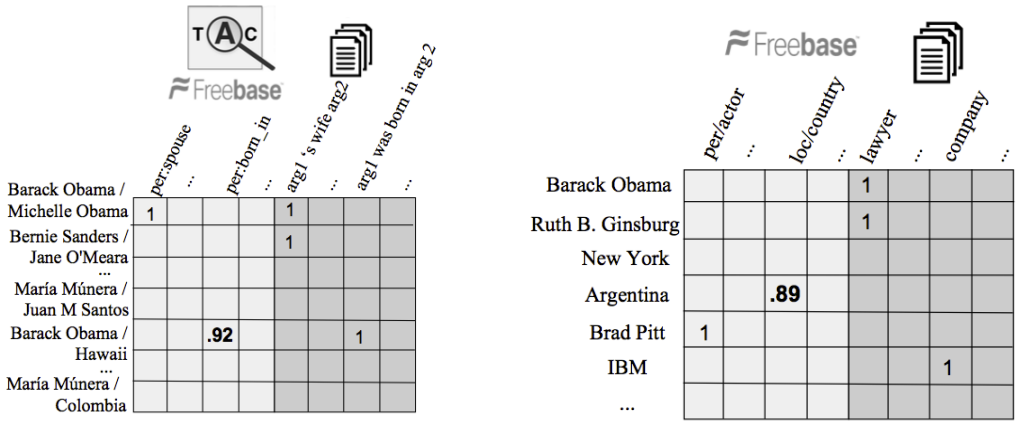


Figure 1. Universal Schema matrix. (Left) Relation extraction. Relation types are represented as columns and entity pairs as rows of a matrix. Both KB relation types and textual patterns from raw text are jointly embedded in the same space. (Right) Entity type prediction. Entity types are represented as columns and entities as rows of a matrix.

In Riedel et al [2], other collaborative filtering approaches are also considered, such as the Neighborhood model [21]:

$$P(y_{(s,r,o)} = 1) = \sigma(\sum_{(s,o,r')\in O-\{(s,o,r)\}} w_{r,r'}) \qquad \textbf{(2)}$$

where $w_{r,r'}$ is a weight representing the association strength between relations *r* and *r'* and O is the set of all observed triples. A model parameterized at the entity level is used as well:

$$P(y_{(s,r,o)} = 1) = \sigma(v_{r,1}^T u_{s,1} + v_{r,2}^T u_{o,2}) \qquad \textbf{(3)}$$

where $v_{r,1}^T$ and $v_{r,2}^T$ are embeddings of the first and second arguments of relation *r* respectively, $u_{s,1}$ is an embedding of entity *s* when it appears as the first argument, *and* $u_{o,2}$ is an embedding of entity *o* when it appears as the second argument.

In these models, we learn vectors in low dimensional embedding spaces for entities, relations (or types), such that similar entities/relations will be near by in embedding space. This naturally allows for reasoning and prediction in the embedding space as well as generalization to unseen triples of entities and relations.

In all cases (including the column-less and row-less models in the following sections), we optimize the models using stochastic gradient descent optimizing the Bayesian Personalized Ranking objective [22]. This is a pairwise ranking objective for optimizing the area under the curve of the relation prediction problem.

3.1.3 Column-less Universal Schema

In the previously described Universal Schema model, each textual pattern is represented as a single embedding, preventing generalization to unseen patterns. This harms its ability to generalize to inputs not precisely seen at training time. To expand the coverage abilities of Universal Schema relation extraction, Verga et al [4] introduce techniques for forming predictions for new relations unseen in training and even for new domains with no associated annotation. In the extreme example of domain adaptation to a completely new language, we may have limited linguistic resources or labeled data such as treebanks, and only rarely a KB with adequate coverage. Our method performs multilingual transfer learning, providing a predictive model for a language with no coverage in an existing KB, by leveraging common representations for shared entities across text corpora. As depicted in Figure 2, we simply require that one language have an available KB of seed facts. We can further improve our models by tying a small set of word embeddings across languages using only simple knowledge about word-level translations, learning to embed semantically similar textual patterns from different languages into the same latent space.

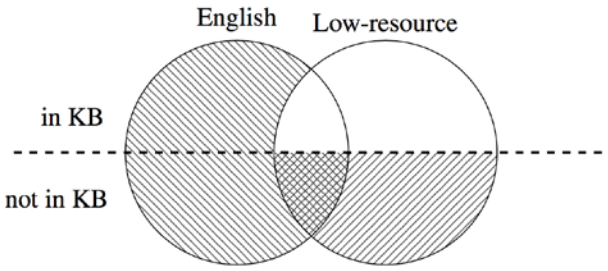


Figure 2. Low-Resource Multilingual AKBC Setting. Splitting the entities in multilingual AKBC training set into parts. We only require that entities in the two corpora overlap. Remarkably, we can train a model for the low-resources language even if entities in the low-resource language do not occur in the KB.

Specifically, Verga et al [4] introduce a *column-less* version of Universal Schema, which uses a *compositional* encoding of textual relations based on recurrent neural networks (RNN) rather than using a single embedding. This allows the model to make predictions about any textual pattern. This parameterization also scales better with the number of textual patterns as the base parameterization is at the word level. The model works by encoding a textual relation (such as … *offered a leadership position at…*) as the last state of an RNN applied to the word level input. The latent factor model objective is then changed to be:

$$P(y_{(s,r,o)} = 1) = \sigma(u_{s,o}^{T} ENCODER(r)) \qquad (4)$$

where *ENCODER(r)* is the output of the LSTM as described. In the Results and Discussion section, we present experiments showing the benefit of this approach for relation extraction in both monolingual and multilingual settings. See Figure 3 for depiction of the model and multilingual setting.

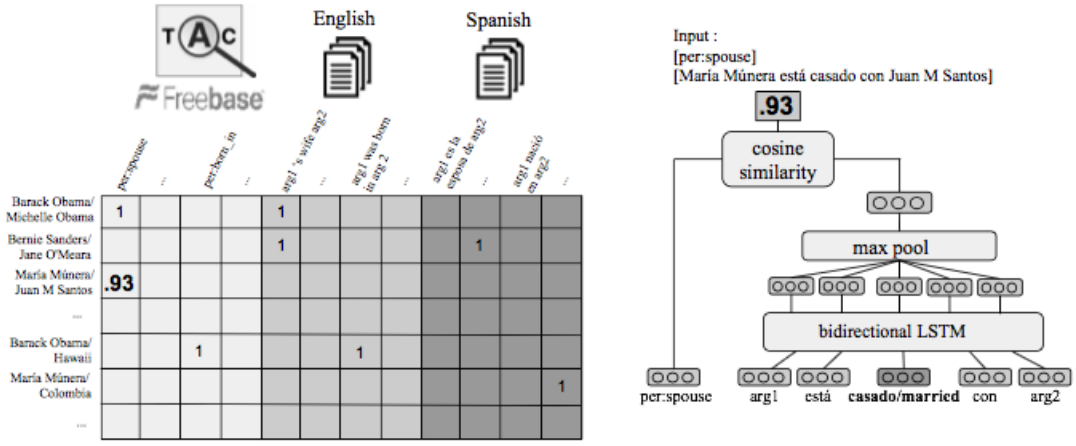


Figure 3. Column-less Universal Schema. Universal Schema jointly embeds KB and textual relations from Spanish and English, learning dense representations for entity pairs and relations using matrix factorization. Cells with a 1 indicate triples observed during training (left). The bold score represents a test-time prediction by the model (right). Using transitivity through KB/English overlap and English/Spanish overlap, our model can predict that a text pattern in Spanish evidences a KB relation despite no overlap between Spanish/KB entity pairs. At train time we use BPR loss to maximize the inner product of entity pairs with KB relations and text patterns encoded using a bidirectional LSTM. At test time we score compatibility between embedded KB relations and encoded textual patterns using cosine similarity. In our Spanish model we treat embeddings for a small set of English/Spanish translation pairs as a single word, e.g. casado and married.

3.1.4 Row-less Universal Schema

In Verga et al [5], we introduce a *row-less* version of Universal Schema. Similar to the neighborhood model, this approach encodes an entity pair by the observed relations of this pair. In this way, we do not store row-specific embeddings. This allows for making predictions for entities or entity-pairs unseen at training time (and hence without a pre-learned row embedding). This work extends the simple neighborhood approach with a variety of encoding functions based on neural network models including some with attention [16][23][24]. These include: *mean-pooling*, in which an entity pair is encoded by the average of its relation vectors, *max-pooling*, in which an entity pair is encoded by the coordinate-wise maximum of its relation vectors, and *attention*, in which a weighted sum over the relation vectors is used. The attention model is specifically defined as, given a query relation which we are trying to predict *c* and entity pair *s,o:*

$$score(r) = v_c^T v_r \; \forall \; r \; s.t. (s,r,o) \in O \quad \textbf{(5)}$$

$$p(r) = exp\,(score(r)) \,/\, \sum_{r' \in \{r' | (s,r',o) \in O\}} exp\,(score(r)) \quad \textbf{(6)}$$

$$ENCODER(s,o) = \sum_{r \in \{(s,r,o) \in O\}} p(r) v_r \quad (7)$$

$$P(y_{(s,r,o)} = 1) = \sigma(ENCODER(s,o)^T v_c) \quad (8)$$

where O is again all observed triples at training time. Figure 4 gives an overview of the row-less model.

In the Results section, we compare the multiple aggregation functions and find that an attention-based aggregation function outperforms several simpler functions and matches a model using explicit row representations with an order of magnitude fewer parameters. More importantly, we also demonstrate that our 'row-less' models accurately predict relations on unseen entity pairs and types on unseen entities.

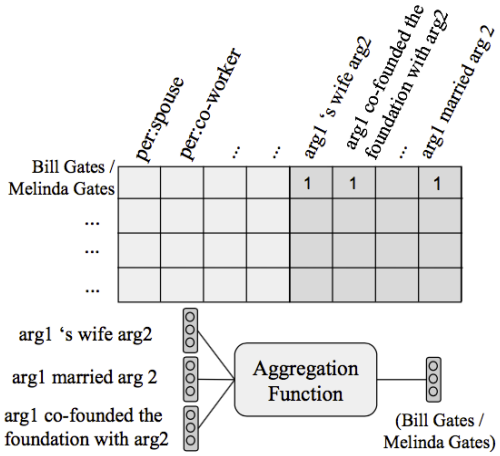


Figure 4. Row-less Universal Schema. Row-less universal schema for relation extraction encodes an entity pair as an aggregation of its observed relation types.

3.1.5 Universal Schema as a Sentence Classifier

A common approach for Sentence Classification is to use distance supervision to align KB facts with supporting evidence in the text, but this is often challenging. For example, not all sentences containing Barack and Michelle Obama state that they are married. A variety of one-shot and iterative methods have addressed the alignment problem [25][26][27][28][29][30][31].

We can circumvent the alignment problem with Universal Schema. As a *sentence classifier* it is meant to predict what knowledge base schema relations a particular sentence encodes. Given a sentence with entities *s* and *o* and free text relation *r*. We use a very simple approach: (1) scan the corpus and extract a large quantity of triplets *(s, r_{text}, o)* where r_{text} is a textual pattern. For each triplet, if the similarity between the embedding of r_{text} and the embedding of a target relation, *r*, is above some threshold, we predict the triplet *(s, r, o)*. In our experiments, we use the cosine distance between the vectors.

3.1.6 Distant Supervision and Search Engine Supervision

In Chang et al [9], we present a method for improving the quality of training data of Universal Schema when used as a sentence classifier. As a baseline approach, we use a distant supervision approach in which entity pairs and relations are extracted from Freebase and filter the textual relations extracted by entity pairs appearing in Freebase. However, this method is not very robust and creates a large amount of noisy training data.

One simple and effective way is asking human annotators to eliminate the noise or manually design noise robust features, and the effectiveness of the approach has been demonstrated in Angeli et al. [33]. However, seeking human annotations is a costly and time-consuming task. Instead, we reduce the noise of our training data by using a search engine in lieu of a human annotator.

For each schema relation (such as those in the TAC-KBP competition schema), we use a few golden manual patterns indicating the relation to select several representative entity pairs. The representative entity pairs mean that if we search the first entity + the golden manual patterns using Google, we should see the second entity very often in the top 10 retrieved web pages. For example, if we search "Barack Obama and his wife", we expect to see many web pages containing "Michelle Obama". This implies (Barack Obama, Michelle Obama) is a good representative entity pair. We extract key phrase candidates using existing distant supervision data. For each representative entity pair and key phrase candidate, we send a query containing the combination of the two, to the search engine, and count the number of mentions of the second entity in the results. The more mentions of the second entity we found, the more likely that this key phrase candidate would imply such relation we want to extract. Finally, we emphasize the textual patterns, which contain at least one verified key phrase candidate, and this can be viewed as softly removing the noises from the training data. An example could be seen in Figure 5. Experimental results presented in the following section validate the effectiveness of this approach over the simple distance supervision baseline.

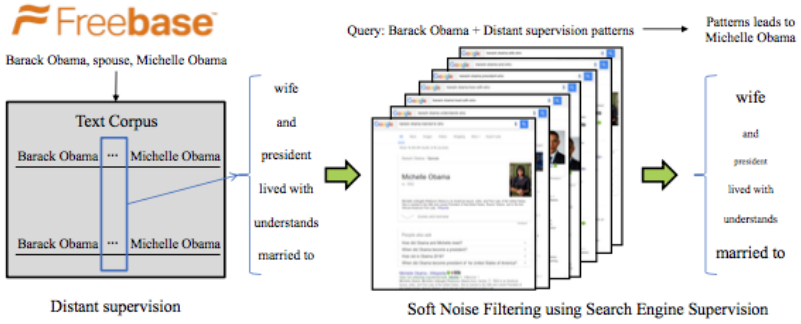


Figure 5. Distant Supervision and Search Engine Supervision. The distant supervision method extracts positive phrases from textual patterns between each entity pair with the relation of interest (per:spouse in this example). However, entity pair might have many different relations other than the ones we are interested in, which consist of the noise in our training data. To alleviate the problem, we examine how often we can find the second entity given the first entity and a positive phrase candidate as the query to a search engine. The noisy patterns would be weighted less in the training data because they usually cannot pass such an inverse check.

3.1.7 Entity Type Prediction

As previously described, we can also use Universal Schema for fine-grained entity type prediction. As is the case in relations, we might observe some entity types in the knowledge base such as *David Ortiz* is a *baseball_player* and the goal is to infer that he might also be a *professional_athlete* but not likely a *senator*. In this case, rows correspond to single entities rather than pairs and columns correspond to entity types. We present experimental results for type prediction.

3.2 Part of Speech Tagging, Parsing, Entity Extraction, and Within Document Coreference

In order to train Universal Schema models and construct knowledge bases from raw text corpora, we perform entity extraction to determine which spans of text refer to entities. This problem, named entity recognition (NER), is typically thought of as a sequence labeling problem, in which tokens are labeled as part of a span of entity with a particular type (e.g. Person, Organization, Location) or not part of an entity span. We describe contributions for efficient and accurate methods for structured prediction/sequence labeling problems including NER and related tasks (part of speech tagging, parsing and within document coreference).

3.2.1 Dynamic Feature Selection

In automatic knowledge base construction, we require the processing of massive corpora from which entities are extracted. Efficiency of entity extraction methods is therefore a key concern. In Strubell et al [7], we present a paired learning and inference algorithm for speeding up the computation time of linear models for common sequence labeling tasks with only a minor degradation in accuracy. This is accomplished by partitioning the features into a sequence of templates, which are ordered such that high confidence can often be reached using only a small fraction of all features. Parameter estimation is arranged to maximize accuracy and early confidence in this sequence. Our approach is simpler and better suited to NLP than other related cascade methods. We present experiments in left-to-right part-of-speech tagging, named entity recognition, and transition-based dependency parsing. On the typical benchmarking datasets we can preserve POS tagging accuracy above 97% and parsing LAS above 88.5% both with over a five-fold reduction in run-time, and NER F1 above 88 with more than 2x increase in speed.

Most of the computation time for these linear models is spent on the dot-product between the sparse features of an example and the weights of the model. In some cases, it is clear that the use of all of these features is excessive and the example can be correctly classified without such features (e.g. in part of speech tagging, labeling “the” as a Determiner or word “hits” preceded by a Cardinal number as a Noun)).

Our method uses *dynamic feature selection*. The model is a linear model consisting of *feature templates* $\{\Phi_j(\boldsymbol{x}, \boldsymbol{y})\}$:

$$\boldsymbol{y}^* = \underset{\boldsymbol{y} \in \boldsymbol{Y}}{\boldsymbol{argmax}} \sum_j \boldsymbol{w_j} \cdot \boldsymbol{\Phi_j}(\boldsymbol{x}, \boldsymbol{y}) \qquad (9)$$

Given a fixed ordering of feature templates, we dynamically decide how many templates should be used in accordance with the model's confidence in the prediction. This procedure is done by building up a *prefix score*, the score of the model up to the i'th template in the ordering:

$$P_{i,y}(x,w) = \sum_{j=1}^{i} w_j \cdot \Phi_j(x,y) \qquad (10)$$

We use $P_{i,y}$ when *x* and *w* are unambiguous. We define P_i to be the vector containing the values $P_{i,y}$ for all *y* in *Y*. With these definitions and a margin parameter *m*, we define *h:*

$$h(P_i, y) = max\{0, \max_{y' != y} P_{i,y'} - P_{i,y} + m\} \qquad (11)$$

The below algorithm boxes describe how *h* is used at training and test time. This function *h* determines whether the model is confident enough to make a prediction on a specific label or if it needs to see more features.

Algorithm 1 Inference

Input: template parameters $\{\mathbf{w}_i\}_{i=1}^{k}$, margin m and optional (for train time) true label y
Initialize: $i = 1$
while $l > 0 \wedge i \leq k$ **do**
  $l = \max_{y'} h(P_i, y')$ (test) or $h(P_i, y)$ (train)
  $i \leftarrow i + 1$
end while
return $\{P_j\}_{j=1}^{i}$ (train) or $\max_{y'} P_{i,y'}$ (test)

Algorithm 2 Parameter Learning

Input: examples $\{(x_i, y_i)\}_i^N$, margin m
Initialize: parameters $\mathbf{w}_0 = 0$, $i = 1$
while $i \leq N$ **do**
  prefixes $\leftarrow$ Infer$(x_i, y_i, \mathbf{w}_i, m)$
  $g_i \leftarrow$ ComputeGradient(prefixes)
  $\mathbf{w}_{i+1} \leftarrow$ UpdateParameters$(\mathbf{w}_i, g_i)$
  $i \leftarrow i + 1$
end while
return $\mathbf{w}_N$

Figure 6. Dynamic Feature Selection Algorithm Boxes

To train this model, we need to both learn the model weights *w* as well as the ordering of the feature templates. The model parameters are trained in such a way to encourage the use of only a few feature templates. This learning procedure can be thought of as learning a separate model per prefix and learning all models simultaneously. Each training example induces an update for a sequence of prefix predictions $\{P_i\}$. Specifically, we have:

$$w^* = \underset{w}{argmin} \sum_{(x,y)} \sum_{i=1}^{i_y^*} h(P_i(x,w), y) \quad (12)$$

where

$$i_y^* = \min_{i \in \{1 \ldots k\}} i \quad s.t. \quad h(P_i, y) = 0 \qquad (13)$$

where *k* is the total number of feature templates. Strubell et al [7] provides additional details on the optimization problem. To learn the ordering of the feature templates, three approaches are presented: (1) Group Lasso [34], in which Group Lasso regularization is applied and the templates are ordered based on the learned regularization weights, (2) Group Orthogonal Matching Pursuit [35][36] in which each feature template performs linear regression to fit the gradient of the los function and templates are added one at a time based on their fit, (3) the Wrapper Method [37], in which templates are added one at a time based on the one with the highest development set performance.

3.2.2 Lexicon Infused Phrase Embeddings for Named Entity Resolution

Semi-supervised information such as word clusters and lexicons is a crucial component of many named-entity recognition approaches. The success of neural-network based language models has led to the use of word representations, embeddings, that effectively capture word semantics. In Passos et al [6], we introduce a new form of learning word embeddings that can leverage information from relevant lexicons to improve the representations, and the first system to use neural word embeddings to achieve (what was at the time) state-of-the-art results on named-entity recognition in both CoNLL and Ontonotes NER.

Lexicons are a simple yet powerful way to provide task-specific supervisory information to the model without the burden of labeling additional data. However, while lexicons have proven useful in various NLP tasks, a small amount of noise in a lexicon can severely impair its usefulness as a feature in log-linear models. For example, even legitimate data, such as the Chinese last name "He" occurring in a lexicon of person last names, can cause the lexicon feature to fire spuriously for many training tokens that are labeled PERSON, and then this lexicon feature may be given low or even negative weight.

We address both these problems by employing lexicons as part of the word embedding training. The Skip-gram model [38][39] can be trained to predict not only neighboring words but also lexicon membership of the central word (or phrase). The resulting embedding training will thus be somewhat supervised by tending to bring together the vectors of words sharing a lexicon membership. Furthermore, this type of training can effectively "clean" the influence of noisy lexicons because even if "He" appears in the PERSON lexicon, it will have a sufficiently different context distribution than labeled named person entities (e.g. a lack of preceding honorifics, etc) that the presence of this noise in the lexicon will not be as problematic as it was previously.

The modified Skip-gram objective in which each word at the center of the context needs to predict its lexicon labels:

$$\underset{\theta}{argmax} \prod_{(w,c) \in D} \; p(c|w;\theta) \; \prod_{l \in Lex} \; p(l|c;\theta)[c \; \in l] \; + \; (1 - p(l|c;\theta))[c \notin l] \qquad (14)$$

where θ are the parameters of the model, *Lex* is the set of all lexicons, (w, c) is a pair of co-occurring words in the corpus *D*, and $[c \; \in l]$ and $[c \notin l]$are indicator functions representing c's membership and nonmembership of the lexicon *l* respectively.

In practice, a very small fraction of words are present in a lexicon-class and this creates skewed training data, with overwhelmingly many negative examples. We address this issue by aggressively sub-sampling negative training data for each lexicon class. We do so by randomly selecting only 1% of the possible negative lexicons for each token.

In this work, we train embeddings for phrases in addition to single tokens. To parse a sequence of tokens into a series of phrases we use a simple greedy approach, which prefers longer phrases to shorter ones. We use PMI to determine a set of bigram phrases. From these bigrams, we select trigrams such that the two bigrams had word overlap.

The learned phrase and lexicon infused embeddings are used as features in an NER model. We scale the embedding vector by a real number, which is a hyper-parameter tuned on the development data. The other features used in the model are similar to those used by Ratinov and Roth [40]: word type (after excluding digits and lowercasing), capitalization pattern, whether it is punctuation, 4-character prefixes and suffixes, character n-grams from length 2 to 5, whether it is in a Wikipedia-extracted lexicon of person names (first, last, and honorifics), dates (months, years), place names (country, US state, city, place suffixes, general location words), organizations, and man-made things, and whether it is a demonym.

Our experimental results perform analysis of the impact of the lexicon infused learned embeddings as well as the other feature groups.

3.2.3 Fast and Accurate Entity Recognition with Iterated Dilated Convolutions

Recently, deep learning methods have been shown to be quite effective in the task of named entity recognition. Bi-directional LSTM have been used for the featurization of sequence text combined with a linear chain Conditional Random Field (Bi-LSTM-CRF) to model dependencies among the output variables [41]. In our work, Strubell et al [8], we present a model that replaces the Bi-directional LSTM with an iterated dilated convolutional neural network (ID-CNN). The ID-CNN captures longer-range dependencies than traditional CNN architectures while requiring fewer parameters. In experimental results, we show that the ID-CNN combined with a CRF is not only more accurate than the Bi-LSTM CRF, but is also faster. The network structure, parameter sharing and training procedures lead to the ID-CNN model with independent predictions to have a 14x speedup over the Bi-LSTM-CRF with only minor degradations in accuracy. The ID-CNN's representational capacity for larger, document-level, contexts can also be used to maintain 8x faster speeds with improved accuracy.

The convolutional neural network (CNN) architectures that we describe are used to generate the input features to the named entity recognition prediction problem. They are differentiable and their parameters are learned in an end-to-end way in the NER task. CNNs are quite commonly used in computer vision, applied to two or more dimensional grids representing pixel values. In modeling natural language, CNNs are typically one-dimensional applied to a sequence of vectors representing tokens. In this setting, a CNN layer is equivalent to applying an affine transformation, *Wc* to a sliding window of width r tokens on either side of each token in the sequence. Mathematically, this is defined as:

$$c_t = W_c \bigoplus_{k=0}^{r} x_{t \pm k} \quad (15)$$

where $x_{t \pm k}$ is the *t'th* token in the sequence, $\bigoplus$ is vector concatenation, producing output c_t. A *dilated* convolution is almost the same operation, except rather than transforming adjacent inputs, the convolution is defined over a wider effective input width by skipping over δ inputs at a time. We refer to δ as the dilation width. The dilated convolution is defined as:

$$c_t = W_c \bigoplus_{k=0}^{r} x_{t \pm k\delta} \quad (16)$$

Notice how similar these two equations are. If the dilation width is set to 1 then the two are equivalent. The dilated convolution incorporates broader context into the representation of a token than a simple convolution using the same number of parameters as a simple convolution with the same radius (i.e. W_c has the same dimensionality).

By *stacking* dilated convolutions together such that each with increasing width, we can capture multiple scales of information, both local information around the token at the center of the convolution as well as global information from a sentence or document. For example, with a radius of 1 and 4 layers of dilated convolutions, the effective input width of each token is width 31, which exceeds the average sentence length (23) in the Penn TreeBank corpus. With a radius of size 2 and 8 layers of dilated convolutions, the effective input width exceeds 1,000 tokens, long enough to encode a full newswire document.

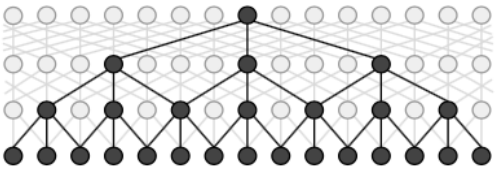


Figure 7. Dilated CNN. A dilated CNN block with maximum dilation width 4 and filter width 3. Neurons contributing to a single highlighted neuron in the last layer are also highlighted.

Unfortunately, simply increasing the depth of stacked dilated CNNs causes considerable overfitting in our experiments. In response, we present Iterated Dilated CNNs (ID-CNNs), which instead apply the same small stack of dilated convolutions multiple times, each iterate taking as input the result of the last application. Repeatedly employing the same parameters in a recurrent fashion provides both broad effective input width and desirable generalization capabilities. We also obtain significant accuracy gains with a training objective that strives for accurate labeling after each iterate, allowing follow-on iterations to observe and resolve dependency violations.

The ID-CNN model operates on a given sequence of T vectors, where the *t'th* vector is x_t (i.e. corresponding to the *t'th* word in the sequence), the model will output a vector of per-entity type prediction scores (or more generally for the sequence labeling per class scores). We refer to this vector of per-class scores as h_t. We use $D_\delta^{(j)}$ to refer to the *j*th dilated convolutional layer of width δ. The first layer of the ID-CNN is a dilation-1 (i.e. normal CNN) layer $D_1^{(0)}$ this defines a representation i_t:

$$i_t = D_1^{(0)} x_t \quad \textbf{(17)}$$

Next L_c layers of dilated convolutions of exponentially increasing dilation width are applied to i_t folding in increasingly broader context into the embedded representation of x_t

$$c_t = r(D_{2^{L_c-1}}^{(j-1)} c_t^{(j-1)}) \quad \textbf{(18)}$$

the final layer is a dilation-1 layer:

$$c_t^{(L_c+1)} = r(D_1^{(L_c)} c_t^{(L_c)}) \quad (19)$$

This stack of dilated convolutions is referred to as a *block* B(.), which has output resolution equal to its input resolution. To incorporate even broader context without over-fitting, we *iteratively* apply *B* L_b times (this introduces no new parameters). Starting with $b_t^{(1)} = B(i_t)$:

$$b_t^{(k)} = B(b_t^{(k-1)}) \quad (20)$$

We apply a simple affine transformation W_o to this final representation to obtain per-class scores for each token x_t :

$$h_t{}^{(L_b)} = W_o b_t^{(L_b)} \quad (21)$$

These features can be used in one of two models for sequence tagging. The first performs independent predictions based on the input sequence, the second uses a linear chain CRF to encode the dependencies among neighboring tokens with the sequence. These two are:

Independent Predictions:

$$P(y|x) = \prod_{t=1}^{T} P(y_t \mid F(x)) \quad (22)$$

CRF:

$$P(y|z) = \frac{1}{Z_x} \prod_{t=1}^{T} \psi_t(y_t|F(x))\, \psi_p(y_t, y_{t-1}) \quad (23)$$

where $F(x)$are the features for the token x, and $\psi_{\square}$ and $\psi_{\square}$ are the CRF potentials.

Rather than optimizing the log likelihood of either of the above equations, we experiment an alternative training method that bridges the gap between these two techniques. Instead of explicit reasoning over output labels during inference, we train the network such that each block is predictive of output labels. Subsequent blocks learn to correct dependency violations of their predecessors, refining the final sequence prediction.

To do so, we first define predictions of the model after each of the L_b applications of the block. Let $h_t{}^{(k)} = W_o b_t^{(k)}$for block k, we minimize the average of the losses for each application of the block:

$$\frac{1}{L_b} \sum_{k=1}^{L_b} \frac{1}{T} \sum_{t=1}^{T} logP(y_t \,|h_t{}^{(k)}) \quad (24)$$

The loss also helps reduce the vanishing gradient problem [42] for deep architectures.

3.2.4 Joint Within Document Coreference

Another important component to knowledge base construction is within document coreference. In this task, we determine which mentions refer to the same entities in the same document. We are concerned with determining coreference between not only named entity mentions but also anaphora.

Coreference resolution systems can benefit greatly from inclusion of global context, and a number of recent approaches have demonstrated improvements when precomputing an alignment to external knowledge sources. However, since alignment itself is a challenging task and is often noisy, existing systems either align conservatively, resulting in very few links, or combine the attributes of multiple candidates, leading to a conflation of entities. Our approach [45] instead performs joint inference between within-document coreference and entity linking, maintaining ranked lists of candidate entities that are dynamically merged and reranked during inference. Further, we incorporate a large set of surface string variations for each entity by using anchor texts from the web that link to the entity.

Instead of fixing the alignment of the mentions to the knowledge base, our proposed approach maintains a ranked list of candidate entities for each mention. To expand the set of surface strings that may be used to refer to each entity, the attributes of each candidate contain anchor texts (the visible text) of the links on the web that refer to that entity candidate. When mentions are compared during inference, we use the features computed from the top ranked entity candidate of the antecedent mention. As mentions are merged, the ranked lists of candidate entities are also merged and reranked, often changing the top-ranked entity candidate used in subsequent comparisons. The large set of surface string variations and constant reranking of the entity candidates during inference allows our approach to correct mistakes in alignment and makes external information applicable to a wider variety of mentions.

We use an approach that does the following: jointly reasons about both within-doc entities and their alignment to KB-entities by dynamically adjusting a ranked list of candidate alignments, during coreference, utilizes a larger set of surface string variations for each entity candidate by using links that appear all over the web [46]. We present results using this approach on the ACE 2004, which outperforms the state-of-the-art at the time coreference system [47] by 0.41 B3 F1 points.

3.2.5 Structured Prediction Energy Networks

Structured Prediction Energy Networks [44][45] are a flexible framework for structured prediction. A deep architecture is used to define an energy function of candidate labels and then predictions are produced by using back-propagation to iteratively optimize the energy with respect to the labels. This deep architecture captures dependencies between labels that would lead to intractable graphical models, and performs structure learning by automatically learning discriminative features of the structured output. One natural application of our technique is multi-

label classification, which traditionally has required strict prior assumptions about the interactions between labels to ensure tractable learning and prediction. We are able to apply SPENs to multi-label problems with substantially larger label sets than previous applications of structured prediction, while modeling high-order interactions using minimal structural assumptions. Overall, deep learning provides remarkable tools for learning features of the inputs to a prediction problem, and this work extends these techniques to learning features of structured outputs. Belanger et al [45] presents end-to-end learning for SPENs, where the energy function is discriminatively trained by back-propagating through gradient-based prediction. In our experience, the approach is substantially more accurate than the structured SVM method [44] as it allows us to use more sophisticated non-convex energies. We provide a collection of techniques for improving the speed, accuracy, and memory requirements of end-to-end SPENs, and demonstrate the power of our method on CoNLL-2005 semantic role labeling tasks. Inexact minimization of non-convex SPEN energies is superior to baseline methods that use simplistic energy functions that can be minimized exactly.

3.3 Entity Resolution

3.3.1 Entity Linking: Resolving to an existing KB

Entity mentions are typically ambiguous. For instance, the surface form "JFK" might refer to the 35th President of the United States, the airport in Queens, the 1991 film, or the library. In order to apply our relation extraction methods, we need to link these ambiguous mentions to unambiguous entity identifiers. We develop methods for resolution to existing knowledge bases (Wikipedia and Freebase) as well as methods for resolution without such an existing knowledge base.

In [9], we present an entity linking system to Wikipedia that is based on word, mention, and entity embeddings. This model uses an embedded representation of a mention context to link to an entity. Three separate representations of contexts and entities are used; these are the local word context, the co-occurring mentions, and the co-occurring entities.

To be clear, the linking model is quite similar to the latent factor model equation in Universal Schema:

$$P(m, C) = \sigma(u_e^T \sum_{c \in C} u_c) \quad (25)$$

where C is the context (either defined as words, mentions, or entities), c is an item of that context and u_e is an entity embedding and $u_c$is a context embedding

To predict entity links, each entity mention is scored. The entity mention with the highest similarity score is linked. Then the remaining mentions are rescored with the entity context updated with the latest link. The similarity score is determined by a simple combination of the (cosine) similarity between the mention contexts and entity representations and the prior probability of the mention referring to a candidate entity as determined by the Wikipedia.

3.3.2 Entity Discovery: Clustering methods for discovering entities

In some cases, we may not have a reference knowledge base to which we want to link ambiguous entity mentions. Entity resolution in this situation is a clustering problem. We want to cluster mentions into groups such that each group refers to a single entity. However, a challenge in applying standard clustering algorithms is the fact that most clustering algorithms do not scale to a large number of clusters (entities), which we expect to be very large in the case of entity resolution. We introduce PERCH [48], a new non-greedy algorithm for online hierarchical clustering that scales to both massive number of points *N* and number of clusters *K*--a problem setting we termed *extreme clustering*. Our algorithm efficiently routes new data points to the leaves of an incrementally-built tree. Motivated by the desire for both accuracy and speed, our approach performs tree rotations for the sake of enhancing subtree purity and encouraging balancedness. We prove that, under a natural separability assumption, our non-greedy algorithm will produce trees with perfect dendrogram purity regardless of online data arrival order. Our experiments demonstrate that PERCH constructs more accurate trees than other tree-building clustering algorithms and scales well with both *N* and *K*, achieving a higher quality clustering than the strongest flat clustering competitor in nearly half the time. While we have not yet applied PERCH to entity resolution datasets at this time, we have applied it to clustering datasets which have similar characteristics in terms of size and number of clusters.

Hierarchical clustering offers two natural advantages to clustering with a large number of clusters. The first is an efficient (typically logarithmic time) way to find nearest neighbors/candidate clusters using a top-down search. The second is the ability to represent multiple alternative clusterings. A hierarchical clustering encodes an array of flat clusterings, known as tree consistent partitions (see Figure 8) [49]. A tree consistent partition is a selection of subtree roots such that the collection of sets of their descendant leaves is a flat clustering. This allows for the delaying of hard clustering decisions until the presence of more data.

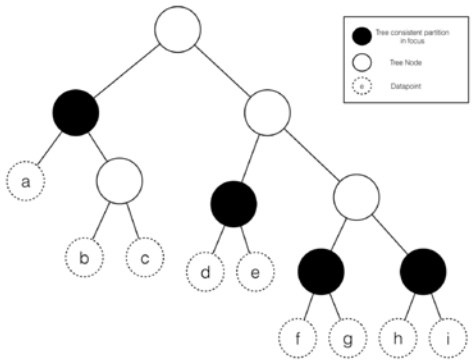


Figure 8. Tree Consistent Partitions. The clustering {{a, b, c}{d, e}{f, g}{h, i}} would be tree consistent with the tree, but {{a, b}{c, d, e}{f, g}} would not. The filled nodes indicate the subtrees that would be selected for the clustering {{a, b, c}{d, e}{f, g}{h, i}}. The clustering {{a, b}{c, d, e}{f, g}, {h, i}} is not consistent because c and d cannot be in the same clustering except in the case where all nodes are in the same cluster

We motivate PERCH by considering the following evaluation metric for hierarchical clustering. Dendrogram purity measures the quality of a hierarchical clustering with respect to a ground truth flat clustering. Dendrogram purity evaluates a hierarchical clustering with respect to all tree

consistent partitions, without requiring a single partition be selected from the tree. Let the ground truth flat clustering be defined as ground truth clustering $C^* = \{C_k^*\}_{k=1}^K$, dividing the dataset into *K* clusters. We use C and C^* as mapping functions from data points to their cluster identities.

Given the pairs of points that are clustered together in the ground truth, P^*, *dendrogram purity* is defined as:

$$DP(T) = \frac{1}{|P^*|}\sum_{k=1}^{K} \sum_{x_i, x_j \in C_k^*} purity(leaves(LCA(x_i, x_j), C_k^*) \quad (26)$$

where *LCA(x,y)* gives the least common ancestor in *T* of *x* and *y* and *purity(set,cluster)* gives the purity of the set with respect to the given cluster label.

We design PERCH to create trees with high dendrogram purity. PERCH stands for Purity Enhancing Rotations for Cluster Hierarchies. The algorithm consists of two stages: nearest neighbor tree insertion and tree maintenance. Data points are added to the PERCH tree one at a time. Each incoming point x_i, we find the leaf node, *l* in *T* that is closest to x_i and perform a *Split* operation on *l*. The *Split* operation, disconnects *l* from its parent *p'*, creates a new leaf node *l'* containing x_i internal node *p* with *l* and *l'* as its children, and adds *p* as a child of *p'*.

Inspired by self-balancing binary search trees, we employ a novel rotation operation that alleviates purity errors caused by a condition we refer to as *masking*. After inserting a new point and creating the leaf *l* with sibling *l'*, the algorithm checks if *l'* is *masked*. If *masking* is detected, a *Rotate* operation is performed, which swaps the positions of *l'* and its aunt in the tree (See Figure 9). After the rotation, the algorithm checks if *l'*s new sibling is masked and recursively applies rotations up the tree. If no masking is detected at any point in the process, the algorithm terminates (see Algorithm box).

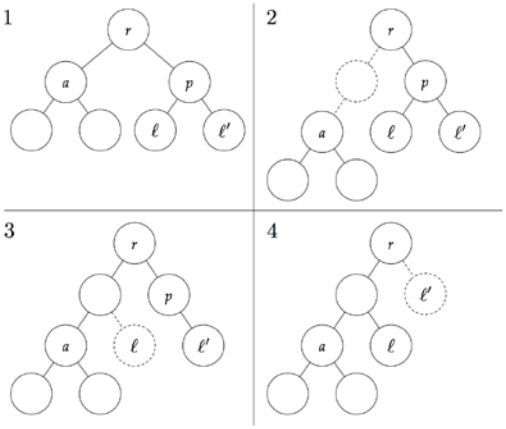


Figure 9. PERCH's Rotation Procedure. The Rotate procedure in four steps. In each step, the change to the tree is indicated by dotted lines.

Masking is defined as: A node *v* with sibling *v'* and aunt *a* in a tree *T* is *masked* if there exists a point *x* in the leaves of *v* such that: $\max_{y \in leaves(v')} || x - y || > \min_{z \in leaves(a)} || x - z ||$. In other words, *v* contains a point *x* that is closer to some point in the aunt *a* than some point in the sibling *v'*.

Intuitively masking happens when a point is misclustered. For example, when a point belonging to the same cluster as *v*'s leaves is sent to *a*, then *v* becomes masked.

Algorithm 1 Insert($x_i, \mathcal{T}$)

t = NearestNeighbor(x_i)
l = Split(t)
for a in Ancestors(l) **do**
  a.AddPt(x_i)
end for
T = T.RotateRec(ℓ.Sibling(), CheckMasked)
T = T.RotateRec(ℓ.Sibling(), CheckBalanced)
if CollapseMode **then**
  T.TryCollapse().
end if
Output: T

Figure 10. PERCH Algorithm Box

We prove that under a natural separation condition on the data, the procedure of nearest neighbor insertions and masking rotations will always create trees with perfect dendrogram purity.

We present the details of using bounding boxes to speed up the computation of nearest neighbor search and the rotation condition in our work [48]. We also present *balance rotations*, which promote more balanced trees for faster nearest neighbor search, *collapsed mode*, which restricts the size of the tree to a given number of leaves, and a *beam* search mode which performs an approximate nearest neighbor search rather than exact.

3.4 Representation Learning

3.4.1 Multi-sense Word Embeddings

Often word vectors are represented by a single vector per word type. However, this ignores polysemy and homonymy, which can be important in downstream tasks using word vectors. In our work [11], we present an extension to the Skip-gram model [38] [39] that efficiently learns multiple embeddings per word type. It differs from recent related work by jointly performing word sense discrimination and embedding learning, by non-parametrically estimating the number of senses per word type, and by its efficiency and scalability. This work produced what were, at the time, state-of-the-art results in the word similarity in context task. Experimental results also demonstrated its scalability by training with one machine on a corpus of nearly 1 billion tokens in less than 6 hours.

As a motivating example of modeling polysemy may be necessary, consider the word type *plant*. In standard Skip-Gram models, *plant*'s embedding is approximately the average of its different contextual semantics relating to biology, placement, manufacturing, and power generation. In moderately high dimensional spaces a vector can be relatively "close" to multiple regions at a time, but this does not negate the unfortunate influence of the triangle inequality here: words that are not synonyms but are synonymous with different senses of the same word will be pulled together.

For example, pollen and refinery will be inappropriately pulled to a distance not more than the sum of the distances *plant–pollen* and *plant–refinery*.

Our approach for modeling sense discriminated vectors is to learn the vectors jointly with the assignment of token contexts to senses; thus we can use the emerging sense representation to more accurately perform the clustering. We also present a non-parametric variant of our method that automatically discovers a varying number of senses per word type.

Specifically, each sense of word has its own embedding. We induce the senses by clustering the embeddings of the context words around each token. The vector representation of the context is the average of its context words' vectors. For every word type, we maintain clusters of its contexts and the sense of a word token is predicted as the cluster that is closest to its context representation. After predicting the sense of a word token, we perform a gradient update on the embedding of that sense. The crucial difference from previous approaches is that word sense discrimination and learning embeddings are performed jointly by predicting the sense of the word using the current parameter estimates. We show that the number of clusters per word type can be made non-parametric. Our approach for this is closely related to online k-means/facility location. We create a new cluster (sense) for a word type with probability proportional to the distance of its context to the nearest cluster (sense).

The below figure compares the architecture of our approach to the standard Skip-gram model:

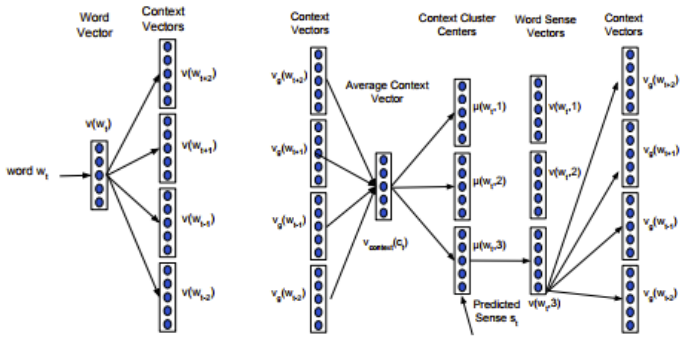


Figure 11. Multi-Sense Skip-gram Architecture. (Left) Architecture of the Skip-gram model with window size $R_t = 2$ Context c_t of word w_t consists of w_{t-1}, w_{t-2}, w_{t+1}, w_{t+2}. (Right) Architecture of MSSG model with window size $R_t = 2$ and $K = 3$. Context c_t of word w_t consists of w_{t-1}, w_{t-2}, w_{t+1}, w_{t+2}. The sense is predicted by finding the cluster center of the context that is closest to the average of the context vectors.

3.4.2 Word Representations via Gaussian Embedding

In most *embedding* models such as Skip-gram and related language models [38][39], Universal Schema [2] and collaborative filtering [21], objects are represented with a single point in vector space. However, this has its limitations. An embedded vector representing a point estimate does not naturally express uncertainty about the target concepts with which the input may be associated. Point vectors are typically compared by dot products, cosine-distance or Euclidean distance, none of which provide for asymmetric comparisons between objects (as is necessary to represent inclusion or entailment). Relationships between points are normally measured by distances required to obey the triangle inequality. Our work [12] advocates moving beyond vector point

representations to potential functions [50] or continuous densities in latent space. In particular, we explore Gaussian embeddings (currently with diagonal covariance), in which both means and variances are learned from data. Gaussians innately represent uncertainty, and provide a distance function per object. KL-divergence between Gaussian distributions is straightforward to calculate, naturally asymmetric, and has a geometric interpretation as an inclusion between families of ellipses. We directly embed words as Gaussian distributional potential functions in an infinite dimensional function space. This allows us to map word types not only to vectors, but to soft regions in space, modeling uncertainty, inclusion, and entailment, as well as providing a rich geometry of the latent space.

We first demonstrate how to fit Gaussian distributions for words *empirically*. We present an energy-based learning framework with two energy functions to directly learn Gaussian embeddings of words. The first is a symmetric function, the expected likelihood or *probability product kernel* [51]. This is the natural extension of the dot product for vector. The second energy function is asymmetric, it is the KL-divergence between two distributions. This even allows us to incorporate more explicit directional supervision re: entailment from a knowledge base or WordNet, is also a sensible objective choice.

Given a pre-trained set of word embeddings trained from contexts, there is a simple way to construct variances using the empirical variance of a word type's set of context vectors.

For a word *w* with *N* word vector sets $\{c(w)_i\}$ representing the words found in its contexts, and window size *W*, the empirical variance is:

$$\Sigma_w = \frac{1}{NW}\sum_i^N \sum_j^W (c(w)_{ij} - w)(c(w)_{ij} - w)^T \quad (27)$$

This is an estimator for the covariance of a distribution assuming that the mean is fixed at *w*. In practice, it is also necessary to add a small ridge term $\delta > 0$ to the diagonal of the matrix to regularize and avoid numerical problems when inverting.

While the dot product between two means of independent Gaussians is a perfectly valid measure of similarity (it is the expected dot product), it does not incorporate the covariances and would not enable us to gain any benefit from our probabilistic model.

The most logical next choice for a symmetric similarity function would be to take the inner product between the distributions themselves. Recall that for two (well-behaved) functions $f, g \in \mathbb{R}^n \rightarrow \mathbb{R}^n$, a standard choice of inner product is

$$\int_{x \in \mathbb{R}^n} f(x)g(x)dx \quad (28)$$

This is the *probability product kernel.* For Gaussians it is*:*

$$\int_{x \in \mathbb{R}^n} N(x; \mu_i, \Sigma_i)\, N(x; \mu_j, \Sigma_j)\, dx = N(0;\, \mu_i - \mu_j, \Sigma_i + \Sigma_j) \quad (29)$$

Since we aim to discriminatively train the weights of the energy function, and it is always positive, we work not with this quantity directly, but with its logarithm. This has two motivations: firstly, we plan to use ranking loss, and ratios of densities and likelihoods are much more commonly worked with than differences – differences in probabilities are less interpretable than an odds ratio.

Secondly, it is easier numerically, as otherwise the quantities can get exponentially small and harder to deal with.

KL-divergence energy function, which is asymmetric, is defined as:

$$-E(P_i, P_j) = D_{KL}(N_j || N_i) = \int_{x \in \mathbb{R}^n} N(x; \mu_i, \Sigma_i) \log \frac{N(x; \mu_j, \Sigma_j)}{N(x; \mu_i, \Sigma_i)} dx$$
$$= \frac{1}{2}(tr(\Sigma_i^{-1}\Sigma_j) + (\mu_i - \mu_j)^T \Sigma_i^{-1}(\mu_i - \mu_j) - d - \log \frac{det(\Sigma_j)}{det(\Sigma_i)}) \quad (30)$$

We use a max-margin loss function for training. We perform L2 regularization on the means. For covariances, we add a hard constraint that the eigenvalues lie within the hypercube $[m,M]^d$ for constants m and M. This requires the covariances to be positive definite as well as reasonably sized.

3.5 Reasoning on Knowledge Graphs

3.5.1 Chains of Reasoning over Entities, Relations, and Text using Recurrent Neural Networks

A common representation for a knowledge base is a graph in which the nodes correspond to entities and edges to relationships. The structure of this graph, or rather, the paths between an entity pair, can be used to predict relationships. This provides a richer input structure than the entity pair alone as is used in the standard Universal Schema setting.

The paths between entities in the knowledge base can be interpreted as Horn clauses. Our work [13] introduces a model based on recurrent neural networks (RNN). This model encodes a path between an entity with an RNN. We extend this model in our work [14]. This extension is presented here. It differs from the original model in that it: (1) learns to jointly reason about relations, entities, and entity-types; (2) uses neural attention modeling to incorporate multiple paths; (3) shares strength in a single RNN that represents logical composition across all relations. As in Universal Schema, the relations that can be encoded in this path include textual patterns as well as knowledge base relations. We refer to the entity pair as (*s*,*o*) and the set of paths between them *S*. This set of paths S is found by using random walks. For a given path, *p* in S and relation *r* about which we are predicting, the model combines the relations along the path sequentially such that the t^{th} relation is given by:

$$h_t = f(W_{hh}h_{t-1} + W_{ih}v_{r_t}) \quad (31)$$

where $W_{hh} \in \mathbb{R}^{kxk}$ and $W_{ih} \in \mathbb{R}^{dxk}$ (d and k are hyperparameters defining the embedding size) and v_{r_t} is the embedding of r_t. The representation v_p we use for p is the last state of the RNN applied to p. We score the similarity between the path p and query relation r by the dot product: $score(p,r) = v_p^T v_r$.

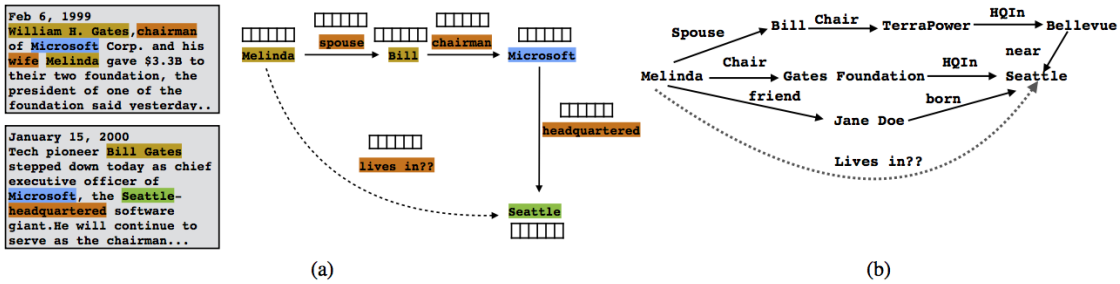


Figure 12. Paths in a Knowledge Graph. The nodes in the knowledge graphs represent entities and the labeled edges represent relations. (a) A path between 'Melinda' and 'Seattle' combining relations from two different documents. (b) There are multiple paths between entities in a knowledge graph. The top two paths are predictive of the fact that Melinda may 'live in' Seattle, but the bottom (fictitious) path is not.

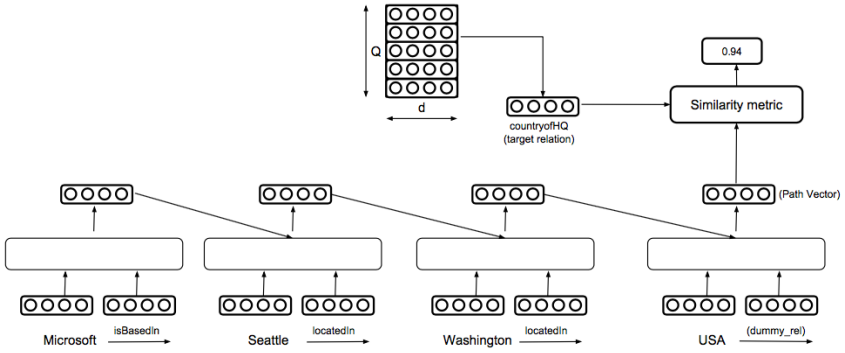


Figure 13. RNN model for Chains of Reasoning. At each step, the RNN consumes both entity and relation vectors of the path. The entity representations can be obtained from its types. The path vector □□ is the last hidden state. The parameters of the final representations of the path and query relation gives a confidence score, with higher scores indicating that the query relation exists between the pair.

Of course, the set of paths, *S*, between *s* and *o* contains multiple alternative paths. For each path, we can compute the score with respect to the query relation. We then experiment with a variety of score pooling functions. These include *maximum* (taking the highest similarity score among the paths), *top-k* (the average of the top k similarity scores), *average* (the average over all paths), and *log-sum-exp* (a smooth approximation to max). The log-sum-exp pooling function is specifically defined as:

$$LSE(p_1, p_2, p_3, \ldots, p_n) = log(\sum_i exp(score(p_i, r))) \quad (32)$$

$$P(r|s,o) = \sigma(LSE(p_1, p_2, p_3, \ldots, p_n)) \; where \; S = \{p_1, p_2, p_3, \ldots, p_n\} \quad (33)$$

The RNN model can be trained in an end-to-end way using backpropagation on triples *(s,r,o)* held out from the knowledge graph. Das et al (2017) also provides a way to incorporate entities and entity types into the model. This extends the equation for a node in a path to be:

$$h_t = f(W_{hh} h_{t-1} + W_{ih} v_{r_t} + W_{eh} v_{e_t}) \quad (34)$$

where $v_{e_t}$is the embedding for entity $e_t$at the position *t* on the path *p* and $W_{eh} \in \mathbb{R}^{mxk}$is a parameter matrix for the entity representations.

Experimental results with this model are presented in the following section. Refer to Figure 12 and 13 for example paths and a depiction of the RNN model.

3.5.2 Learning a Natural Language Interface with Neural Programmer

Ultimately, we would like to be able to perform reasoning more sophisticated than predicting binary relations between pairs of entities. We would like to support a natural language interface in which questions requiring complicated, multi-step inference on knowledge bases, can be answered. The task, which involves both language understanding and reasoning, is often approached by mapping natural language queries to logical forms or programs that provide the desired response when executed on the database. Our work [15] presents to our knowledge, the first weakly supervised, end-to-end neural network model to induce such programs on a real-world dataset. We enhance the objective function of Neural Programmer [16] a neural network with built-in discrete operations, and apply it on WikiTableQuestions, a natural language question-answering dataset. The model is trained end-to-end with weak supervision of question-answer pairs, and does not require domain-specific grammars, rules, or annotations that are key elements in previous approaches to program induction. We present experimental results which are competitive to the current state-of-the-art accuracy obtained by a traditional natural language semantic parser.

Neural Programmer is a neural network augmented with discrete operations. There are four components to the model. (1) A Question Recurrent Neural Network learns a representation of the query to the natural language interface. (2) A list of discrete operations that are manually defined. Each operation is represented by a real valued embedding vector that is learned during training. These operations include things like counting, min/max operations, selection, etc. (3) A selector module represents a probability over the operations defined in (2) as well as a probability distribution over the columns of the database table. (4) A history RNN in which each timestep of the network is an observation of the probability distributions over operations and columns from (3). This component produces the input to the selector in (3).

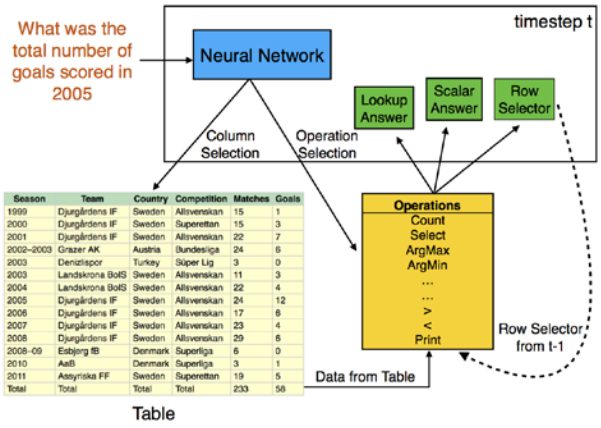


Figure 14. Neural Programmer. Neural Programmer is a neural network augmented with a set of discrete operations. The model runs for a fixed number of time steps, selecting an operation and a column from the table at every time step. The induced program transfers information across timesteps using the row selector variable while the output of the model is stored in the scalar answer and lookup answer variables.

3.6 FACTORIE Machine Learning and Natural Language Processing Toolkit

FACTORIE is a probabilistic modeling toolkit with scalable and effective machine learning and natural language processing capabilities developed by McCallum and his IESL lab. As a part of this program, we have developed numerous improvements to this Scala library. These include: implementations of Universal Schema training using Bayesian Personalized Ranking, Universal Schema sentence classification tools, English tokenization with JFlex (50x faster, ~500k tokens/second), Chinese word segmentation and POS tagging, advancing work towards Arabic tokenization, ~2x faster dependency parser, significant NER improvements for English and Spanish, generalized MCMC hierarchical entity resolution module, multiple efficient word embedding training algorithms with ranking objectives, and more. We also offer open source implementations of nearly all of the work described here including: Column-less and Row-less Universal Schema, Dilated CNNs for NER, extreme clustering and PERCH, Chains of Reasoning, Neural Programmer, and more.

4.0 RESULTS AND DISCUSSION

4.1 Relation Prediction using Universal Schema

In our work [2], we perform an information retrieval based evaluation of relation prediction with Universal Schema. We perform evaluation on both prediction of Freebase relations and textual patterns. Each relation is treated as a query and entity pairs satisfying that relation as the search results. For each relation, we retrieve the top 1000 entity pairs and pool answers from the top 100 answers from each system for manual evaluation of their relevance. Performance is measured for each relation by *average precision* and the mean average precision (MAP) across all relations is computed as the overall metric. We also report the MAP score in which average precisions are weighted by the number of true relations in each calculation. We compare the performance of each of the Universal Schema models (latent factor, neighborhood, and entity-based) and their combinations to three baseline methods: the distant supervision based classifier of Mintz et al [26]; its extension, Yao et al [28]; and the Multi-Instance Multi-Label system [30], which was state of the art at the time of this experiment.

Following previous work [27], our documents are taken from the NYTimes corpus [52]. Articles after 2000 are used as training corpus, articles from 1990 to 1999 as test corpus. We also split Freebase facts 50/50 into train and test facts, and their corresponding tuples into train and test tuples. Then we align training tuples with the training corpus, and test tuples with the test corpus. This alignment relies on a preprocessing step that links NER mentions in text with entities in Freebase. In our case we use a simple string-match heuristic to find this linking. Now we align an entity tuple (t_1, t_2) with a pair of mentions (m_1, m_2) in the same sentence if m_1 is linked to t_1 and m_2 to t_2. Based on this alignment we filter out all relations for which we find fewer than 10 tuples with mentions in text. The above alignment and filtering process reduces the total number of tuples related according to Freebase to 16k: approximately 8k tuples with facts mentioned in the training set, and approximately 8k such tuples for the test set. In addition, we have a set of approximately 200k training tuples for which both arguments appear in the same sentence and both can be linked to Freebase entities, but for which no Freebase fact is recorded. This can either be because they are not related, or simply because Freebase does not contain the relationship yet. We also have about 200k such tuples in the test set. To simplify evaluation, we create a subsampled test set by randomly choosing 10k of the original test set tuples. We use lexicalized dependency paths in place of free text relations. The observed facts are the union of lexicalized dependency path relations and freebase facts.

Table 1 represents results on Freebase relations and Table 2 presents results on text patterns. Our systems are labeled by neighborhood model (N), the factorized model (F), their combination (NF) and the combined model with a latent entity representation (NFE). For all our models we use the same number of components when applicable ($K^F = K^E = 100$), 1000 epochs, and 0.01 as regularizer for component weights and 0.1 for neighborhood weights. Table 1 shows that adding pattern cluster features (and hence incorporating more data) helps YA11 to improve over MI09. Likewise, we see that the factorized model F improves over N, again learning from unlabeled data. This improvement is bigger than the corresponding change between MI09 and YA11, possibly indicating that our latent representations are optimized directly towards improving prediction performance. The combination of N, F and E outperforms all other models in terms of weighted

MAP, indicating the power of selectional preferences learned from data. Note that NFE is significantly different ($p << 0.05$ in sign test) to all but the NF and F models. In terms of MAP, the NF model outperforms NFE, indicating that it does not do as well for frequent relations, but better for infrequent ones. In Table 2, we evaluate the Universal Schema models on text patterns. We again see that learning a latent representation (F, NF and NFE) from additional data helps quite substantially over the N model.

Table 1. Averaged and weighted MAP for Freebase Relations on Pooled Results

Relation	#	MI09	YA11	SU12	N	F	NF	NFE
person/company	103	0.67	0.64	0.70	0.73	0.75	0.76	**0.79**
location/containedby	74	0.48	0.51	0.54	0.43	0.68	0.67	**0.69**
author/works_written	29	0.50	0.51	0.52	0.45	0.61	0.63	**0.69**
person/nationality	28	0.14	**0.40**	0.13	0.13	0.19	0.18	0.21
parent/child	19	0.14	0.25	0.62	0.46	0.76	**0.78**	0.76
person/place_of_death	19	0.79	0.79	0.86	**0.89**	0.83	0.85	0.86
person/place_of_birth	18	0.78	0.75	0.82	0.50	0.83	0.81	**0.89**
neighborhood/neighborhood_of	12	0.00	0.00	0.08	0.43	0.65	0.66	**0.72**
person/parents	7	0.24	0.27	0.58	0.56	0.53	**0.58**	0.39
company/founders	4	0.25	0.25	0.53	0.24	0.77	**0.80**	0.68
film/directed_by	4	0.06	0.15	0.25	0.09	0.26	0.26	**0.30**
sports_team/league	4	0.00	0.43	0.18	0.21	0.59	**0.70**	0.63
team/arena_stadium	3	0.00	0.06	0.06	0.03	0.08	**0.09**	0.08
team_owner/teams_owned	2	0.00	0.50	0.70	0.55	0.38	0.61	**0.75**
roadcast/area_served	2	*1.00*	0.50	*1.00*	0.58	0.58	0.83	*1.00*
structure/architect	2	0.00	0.00	*1.00*	0.27	*1.00*	*1.00*	*1.00*
composer/compositions	2	0.00	0.00	0.00	0.50	0.67	**0.83**	0.12
person/religion	1	0.00	*1.00*	*1.00*	0.50	*1.00*	*1.00*	*1.00*
film/produced_by	1	*1.00*	*1.00*	*1.00*	*1.00*	0.50	0.50	0.33
MAP		0.32	0.42	0.56	0.45	0.61	0.66	0.63
Weighted MAP		0.48	0.52	0.57	0.52	0.66	0.67	0.69

The # column shows the number of true facts in the pool. NFE is statistically different to all but NF according to the sign test. Bold faced are winners per relation, italics indicate ties.

Table 2. Averaged and weighted MAP on Surface Patterns

Relation	#	N	F	NF	NFE
visit	80	0.19	**0.68**	0.49	0.42
attend	69	**0.23**	0.10	0.07	0.10
base	61	0.46	**0.87**	0.81	0.68
head	38	0.47	0.67	**0.70**	0.68
scientist	36	0.25	**0.84**	0.79	0.73
support	18	0.16	0.29	0.32	**0.38**
adviser	11	0.19	0.15	0.19	**0.28**
criticize	9	0.09	0.60	**0.67**	0.64
praise	4	0.01	0.03	0.05	**0.10**
vote	3	0.18	0.18	**0.34**	0.34
MAP		0.22	0.44	0.44	0.43
Weighted MAP		0.28	0.56	0.50	0.46

4.2 Column-less Universal Schema Slot Filling Evaluation

We begin by presenting results using column-less Universal Schema to predict *is-a* relationships on a number of languages. We compare Universal Schema (multitasking) to individual classifiers:

Table 3: *is-a* Relation Prediction with Column-less Universal Schema

Language	F1 (classifier)	F1 (multitasking)
ES	.915	**.933**
DE	.905	**.923**
FR	.909	**.927**
IT	.915	**.933**
PT	.919	**.936**
JP	.908	**.922**
RU	.913	**.933**

Next, in the slot filling task, a system is given a triple consisting of an entity pair and relation *(s,r,o)* triple with one of the entities held out. The task is fill in as many correct values for the held-out entity as possible. For example, if the query triple were (*Michelle Obama*, *per:married-to*, <blank>), we would want to predict *Barack Obama*, and if the query triple were (*Seattle*, *gpe:headquarters-in*, <blank>) with *Microsoft*, *Amazon*, and the slew of the other companies satisfying this.

In the TAC-KBP slot filling tasks, an evaluation set is collected by having annotators label the responses from multiple systems as correct or not correct. Evaluation is done by measuring the precision/recall/F1 of the response predicted by our system compared to this gold set.

In these experiments, we compare the standard Universal Schema model to the column-less/compositional model. We evaluate on three different slot filling tasks: English TAC 2013, English TAC 2014, and Spanish TAC 2012. As a baseline we use a CNN, using width-3 convolutions, followed by tanh and max pool layers [53].

In these experiments, we see the effectiveness of the column-less Universal Schema model as it outperforms the standard approach. We also see that in the ensemble setting the system is able to outperform even the top performing system in the challenge. In experiments on the English and Spanish TAC KBC slot-filling tasks, we find that both Universal Schema and column-less (called "LSTM") models outperform the CNN across languages, and that the LSTM tends to perform slightly better than standard Universal Schema as the only model. Ensembling the LSTM and standard Universal Schema models further increases final F1 scores in all experiments, suggesting that the two different types of model complement each other well. The English results on the 2013 and 2014 slot filling task are shown in Table 4.

Table 4. TAC-KBP English 2013 & 2014 Slot Filling Results

Model	Recall	Precision	F1
CNN	31.6	36.8	34.1
LSTM	32.2	39.6	**35.5**
USchema	29.4	42.6	34.8
USchema+LSTM	34.4	41.9	37.7
USchema+LSTM+Es	38.1	40.2	**39.2**
USchema+LSTM+AN	36.7	43.1	39.7
USchema+LSTM+Es+AN	40.2	41.2	**40.7**
Roth et al. (2014)	35.8	45.7	40.2

Model	Recall	Precision	F1
CNN	28.1	29.0	28.5
LSTM	27.3	32.9	**29.8**
USchema	24.3	35.5	28.8
USchema+LSTM	34.1	29.3	31.5
USchema+LSTM+Es	34.4	31.0	**32.6**

AN refers to an alternative names heuristic and Es refers to the addition of Spanish text at train time. (Left) 2013 Results LSTM+USchema ensemble outperform any single model including the highly tuned top 2013 system of Roth et al (2014) despite using no handwritten patterns. (Right) 2014 Results. The AN heuristic is ineffective on 2014 data, adding only 0.2 on F1. Our system would rank 4/18 in the official TAC2014 competition behind systems that use hand-written patterns and active learning despite using neither of these additional annotations (Surdeanu and Ji., 2014).

The LSTM and standard Universal Schema models each perform better on different pattern lengths and are characterized by different precision-recall tradeoffs as seen in Table 5 and 6. We compare the precision / recall curve for the TAC-2013 and observe that the compositional model (LSTM) is able to achieve better recall than the standard approach (USchema), while the standard approach achieves better precision. We also notice that the LSTM performs better on longer patterns whereas USchema performs better on shorter patterns.

Table 5. Precision/Recall Curve for LSTM & USchema on 2013 TAC SF

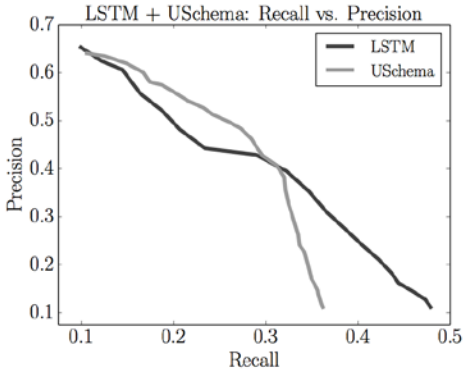
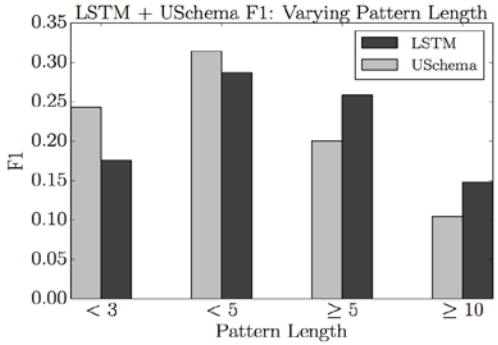


Table 6. F1 by Pattern Length on 2013 TAC SF



We also show the robustness of the system in a setting without labeled data in Table 7. We evaluate on a zero-shot transfer learning task on the Spanish 2012 slot filling task. In this setting only English knowledge base relations and English and Spanish textual patterns are used. We see that adding a translation dictionary and ensembling models improves results.

Table 7. Zero-annotation transfer learning F1 scores on Spanish 2012 TAC

Model	Recall	Precision	F1
LSTM	9.3	12.5	10.7
LSTM+Dict	14.7	15.7	15.2
USchema	15.2	17.5	16.3
USchema+LSTM	21.7	14.5	17.3
USchema+LSTM+Dict	26.9	15.9	**20.0**

4.3 Row-less Universal Schema Experiments

In our work [5], we compare the row-less universal schema model to a model based on explicit row embeddings and compositional universal schema. We evaluate our models on a relation extraction task using the FB15K-237 dataset [53]. The data is composed of a small set of 237 Freebase relations and approximately 4 million textual patterns from Clueweb with entities linked to Freebase [54]. In past studies, for each (subject, relation, object) test triple, negative examples are generated by replacing the object with all other entities, filtering out triples that are positive in the data set. The positive triple is then ranked among the negatives. In our experiments we limit the possible generated negatives to those entity pairs that have textual mentions in our training set. This way we can evaluate how well the model classifies textual mentions as Freebase relations. We also filter textual patterns with length greater than 35. Our filtered data set contains 2740237 relation types, 2014429 entity pairs, and 176476 tokens. We report the percentage of positive triples ranked in the top 10 amongst their negatives as well as the MRR scaled by 100.

Models are tuned to maximize mean reciprocal rank (MRR) on the validation set with early stopping. The entity pair model used a batch size 1024, l2 = 1e-8, ε = 1e-4, and learning rate 0.01. The aggregation models all used batch size 4096, l2 = 0, ε = 1e-8, and learning rate 0.01. Each model uses 200 negative samples except for max pool, which performed better with two negative samples. The column vectors are initialized with the columns learned by the entity pair model. Randomly initializing the query encoders and tying the output and attention encoders performed better and all results use this method. All models are trained with embedding dimension 25.

Our results are shown in Table 8. We can see that the models with query specific aggregation functions give the same results as models with explicit entity pair representations. The Max Relation model performs competitively with the Attention model, which is not entirely surprising as it is a simplified version of the Attention model. Further, the Attention model reduces to the Max Relation model for entity pairs with only a single observed relation type. In our data, 64.8% of entity pairs have only a single observed relation type and 80.9% have 1 or 2 observed relation types.

We also explore the models' abilities to predict on unseen entity pairs (Table 9). We remove all training examples that contain a positive entity pair in either our validation or test set. We use the same validation and test set as in Table 8. The entity pair model predicts random relations as it is unable to make predictions on unseen entity pairs. The query-independent aggregation functions, mean pool and max pool, perform better than models with explicit entity pair representations. Again, query specific aggregation functions get the best results, with the Attention model performing slightly better than the Max Relation model.

In Tables 10 and 11, we repeat this experiment incorporating *column-less* or *compositional* Universal Schema producing an approach that can generalize to both unseen rows and columns. While both the MRR and Hits@10 metrics increase for models with explicit row representations, the row-less models show an improvement only on the Hits@10 metric. The MRR of the query dependent row-less models is still competitive with the model with explicit row representation even though they have far fewer parameters to fit the data.

The two experiments indicate that we can train relation extraction models without explicit entity pair representations that perform as well as models with explicit representations. We also find that models with query specific aggregation functions accurately predict relations for unseen entity pairs.

Table 8. Row-less Universal Schema MRR and Hits@10 on FB15K-237

Model	MRR	Hits@10
Entity-pair Embeddings	31.85	51.72
Mean Pool	25.89	45.94
Max Pool	29.61	49.93
Attention	31.92	51.67
Max Relation	31.71	51.94

The percentage of positive triples ranked in the top 10 amongst their negatives as well as the mean reciprocal rank (MRR) scaled by 100 on a subset of the FB-15K-237 dataset. All positive entity pairs in the evaluation set are unseen at training time. Entity-pair embeddings refer to the model with explicit row representations.

Table 9. Row-less Universal Schema on FB15K-237 on Unseen Entity Pairs

Model	MRR	Hits@10
Entity-pair Embeddings	5.23	11.94
Mean Pool	18.10	35.76
Max Pool	20.80	40.25
Attention	**29.75**	49.69
Max Relation	28.46	48.15

In this table, we are predicting entity pairs that are not seen at train time.

Table 10. Row-less & Column-less Universal Schema on FB15K-237

Model	MRR	Hits@10
Entity-pair Embeddings	31.85	51.72
Entity-pair Embeddings-LSTM	**33.37**	54.39
Attention	31.92	51.67
Attention-LSTM	30.00	53.35
Max Relation	31.71	51.94
Max Relation-LSTM	30.77	**54.80**

Negative examples are restricted to entity pairs that occurred in the KB or text portion of the training set. Models with -LSTM are column-less.

Table 11. Row-less & Column-less Universal Schema on FB15K-237 on Unseen Entity Pairs

Model	MRR	Hits@10
Entity-pair Embeddings	5.23	11.94
Attention	**29.75**	49.69
Attention-LSTM	27.95	51.05
Max Relation	28.46	48.15
Max Relation-LSTM	29.61	**54.19**

4.4 Entity Type Prediction

We also apply Universal Schema to entity type prediction [3] [5]. We extract entity types from New York Times data for the years 1990 to 2007 [52] using the dependency parse of sentences. We extract paths up to and including a verb (such as … *buy share* and … *roll over*) and appositive structures (such as …, *co-founder*, ...) [3]. Since there is no ground truth for these patterns (other than a subsample of positive-only cells), we cannot easily evaluate them. We query some of them and list the top ranked entities according to our model and the baseline system and have annotators hand labeled relevance. Entities from NYT articles are split into training (355,942 entities) and test set (147,359 entities). In total the input matrix has approximately 500K rows (entities), 17K columns (types). When training the model we hide all query patterns in the test set. We compare our approach against a binary classifier (Marked "Classifier" in table) that considers entities co-occurring with the query pattern as positive examples and all others as negative examples. As the classification model we use maximum entropy. We also compare against the multi-instance multi-label distant supervision classifier for fine-grained entity recognition (marked UW) [55]. We measure precision, recall and F1 for entities in this set. For each entity, patterns with probabilities above a threshold are considered as true. The threshold is set to 0.5. In scenarios where an entity has no patterns above the threshold, the top ranked one is selected. This may lower the precision of each system, however, it does increase the recall and F1 score for both our approach and the baseline. Table 12 lists the F1 measures for each pattern. We can see that our approach performs significantly better than the baseline on 6 out of 8 patterns. We perform slightly worse on two patterns. On micro average, we gain approximately 15% in F1 score against the classifier, 7% against UW. When analyzing the errors made by the baseline system we see most problems when there are no patterns above the threshold. In these cases the baseline's top ranked patterns (now with score under the threshold) are mostly incorrect. However, for our model the top ranked patterns, when under the threshold, are still often correct.

Table 12. Universal Schema for Entity Type Prediction on Text Patterns

Query	Classifier	UW	Universal
politician	0.448	0.549	**0.738**
scientist	0.354	0.404	**0.499**
magnate	0.460	**0.529**	0.433
band	**0.427**	0.404	0.413
reporter	0.330	0.356	**0.437**
actor	0.518	0.598	**0.649**
player	0.711	0.794	**0.840**
magazine	0.675	0.727	**0.845**
Overall	0.557	0.635	**0.701**

F1 measure on 8 patterns.

In our Row-less Universal Schema work [5], we also evaluate on entity type prediction. In these experiments, we collect all entities along with their types from a dump of Freebase. We then filter all entities with less than five Freebase types, leaving a set of 844780 (entity, type) pairs. Additionally, we collect 712072 textual (entity, type) pairs from Clueweb. The textual types are the 5000 most common appositives extracted from sentences mentioning entities. This results in 140513 unique entities, 1120 Freebase types, and 5000 free text types. All embeddings are 25 dimensions, randomly initialized. We tune learning rates from {.01, .001}, l2 from {1e-8, 0}, batch size {512, 1024, 2048} and negative samples from {2, 200}. For evaluation, we split the Freebase (entity, type) pairs into 60% train, 20% validation, and 20% test. We randomly generate 100 negative (entity, type) pairs for each positive pair in our test set by selecting random entity and type combinations. We filter out false negatives that were observed true (entity, type) pairs in our complete data set. Each model produces a score for each positive and negative (entity, type) pair where the type is the query. We then rank these predictions, calculate average precision for each of the types in our test set, and then use those scores to calculate mean average precision (MAP). Table 13 shows the results of this experiment. We can see that the query dependent aggregation functions (Attention and Max Relation) performs better than the query independent functions (Mean Pool and Max Pool). The performance of models with query dependent aggregation functions which have far fewer parameters match the performance of the model with explicit entity representations. Table 14 shows the results of the experiment with unseen entities. There is very little performance drop for models trained with query dependent aggregation functions. The performance of the model with explicit entity representations is close to random.

Table 13. Row-less Universal Schema for Entity Type Prediction

Model	MAP
Entity Embeddings	54.81
Mean Pool	39.47
Max Pool	32.59
Attention	**55.66**
Max Relation	**55.37**

Table 14. Row-less Universal Schema for Entity Type Prediction on Unseen Entities

Model	MAP
Entity Embeddings	3.14
Mean columns	34.77
Max column	43.20
Mean Pool	35.53
Max Pool	30.98
Attention	**54.52**
Max Relation	**54.72**

4.5 TAC-KBP 2016 Results

In this section, we present the results from our TAC-KBP Cold Start system in 2016 [9]. These results include many of algorithms and methods described in this report, including the compositional Universal Schema model as well as the embedding based entity-linking system. We participated in 3 language settings: English only, Spanish only, and the both languages.

In these experiments, we evaluated two modes of the system. In the first, the system is used in a *slot filling* evaluation task in which queries are given to the system and entity expansion and retrieval decisions can be made on the fly. In the second, the system is used to construct a knowledge base (KB) and queries are answered programmatically through KB lookups. In each settings, the slot filling queries can consist of either one or two "hops". The single hop setting is identical to the evaluation task presented in the Columnless Universal Schema Experiments Section. In the two hop setting, we are given an ordered pair of triples. The first triple contains a missing object entity and the second triple has both the predicate and object entity. Evaluation is performed on the predicted object entities of the second tuple. For example, the two hop query might be: (*Barack Obama, per:married-to,<blank>) (* <blank>, per:born-in, <blank>) which asks the question where was Barack Obama's spouse born?

We evaluate several configurations of the system: with and without search engine supervision/distance supervision date, compositional / non-compositional Universal Schema. The below tables summarize the system settings and results for each setting.

In Table 15, the English hop0 scores show that search engine supervision (SF5 ENG and KB4 ENG) have higher precision and similar F1 compared with the distant supervision (SF4 ENG and KB2 ENG). Furthermore, merging the results (SES+DS) of these sentence classifiers lead to a better F1 in hop0 (SF1 ENG and KB1 ENG). For the slot filling task, the within document expansion using within-document coreference also improves the F1 overall (from 0.1693 in SF2 ENG to 0.1740 in SF1 ENG).

In Table 16, we can see that within document query expansion boosts the performance a little bit by comparing the SF1 SPA and SF2 SPA. However, adding the alias information from redirect page in Wikipedia seems to hurt the performance in Spanish by comparing the SF1 SPA and SF4 SPA. It would need further investigation to know the reason. It is difficult to tune the thresholds for Spanish sentence classifiers because we don't have annotations for the inverse relations and

hop1 responses in the TAC 2012 and Spanish pilot runs for 2016. Thus, we prepare 2 different thresholds and skip all hop1 queries in the SF task. The first way is to disallow any threshold to go below 0.25, and multiply all the resulting thresholds by 0.7. The second way is to disallow any threshold to be lower than 0.1. The results indicate that the former method gives higher performance. The classifier thresholds used for our KB and SF are roughly the same. However, our KB gets high precision but low recall results, while our SF generates high recall but low precision responses. This could indicate that either our Spanish query expansion generates too many noisy aliases or the Spanish entity linking over-split the discovered entities. It was on the Spanish track that our system was ranked first among the competitors.

In 2016, we only used the English and Spanish corpus and queries, but correct Chinese responses from other teams are also considered while computing the recall. This explains the generally low recall in Table 17. In most of the submissions, we just perform cross-lingual entity linking or query expansion in a English run and a Spanish run, and the method column in Table 17 indicates which runs we choose. The KB4 XLING and KB5 XLING are exceptions because we want to compare the performances of only using universal schema and only using LSTM, while all other runs are the combination of these two sentence classifiers. By comparing these runs, we see that the LSTM (KB5 XLING) has much higher recall than universal schema (KB4 XLING). The combination of these two with the search engine supervision can further improve the performance (e.g., KB3 XLING).

Table 15. TACKBP 2016 English SF & KB Results

Run	Method	Description
SF1_ENG	SES + DS	Merging the results w/ and w/o Search Engine Supervision
SF2_ENG	SES + DS + No_Doc_Exp	As run SF1_ENG, skipping within document expansion
SF3_ENG	2015 System	The best UMass_IESL 2015 SF system
SF4_ENG	DS	Distant Supervision (i.e., w/o Search Engine Supervision)
SF5_ENG	SES	Search Engine Supervision
KB1_ENG	SES + DS	Merging the results w/ and w/o Search Engine Supervision.
KB2_ENG	DS	Distant Supervision (i.e., w/o Search Engine Supervision)
KB3_ENG	2015 System	The best UMass_IESL 2015 KB system
KB4_ENG	SES	Search Engine Supervision
KB5_ENG	KB1_ENG, ENG_Th3	As run KB1_ENG, but only tuning the thresholds by optimizing the F1 in annotated samples.

LDC max	hop0			hop1			All		
Run	**Prec**	**Rec**	**F1**	**Prec**	**Rec**	**F1**	**Prec**	**Rec**	**F1**
SF1_ENG	0.2288	**0.2026**	**0.2149**	0.4444	0.0130	0.0252	0.2323	**0.1391**	**0.1740**
SF2_ENG	0.2442	0.1895	0.2134	0.1304	0.0195	0.0339	0.2342	0.1326	0.1693
SF3_ENG	**0.3023**	0.1536	0.2037	0.0811	**0.0877**	**0.0842**	0.1879	0.1315	0.1547
SF4_ENG	0.2441	0.1683	0.1992	0.5000	0.0130	0.0253	0.2488	0.1163	0.1585
SF5_ENG	0.2893	0.1503	0.1978	**0.5714**	0.0130	0.0254	**0.2954**	0.1043	0.1542
KB1_ENG	0.2709	0.1536	**0.1960**	0.0392	0.0714	0.0506	0.1278	0.1261	0.1269
KB2_ENG	0.3008	0.1209	0.1725	0.0539	0.0519	0.0529	0.1657	0.0978	0.1230
KB3_ENG	**0.3900**	0.1275	0.1921	**0.2526**	0.0779	**0.1191**	**0.3458**	0.1109	**0.1679**
KB4_ENG	0.3004	0.1193	0.1708	0.1170	0.0649	0.0835	0.2246	0.1011	0.1394
KB5_ENG	0.2110	**0.1634**	0.1842	0.0191	**0.0942**	0.0317	0.0647	**0.1402**	0.0886

Table 16. TACKBP 2016 Spanish SF & KB Results

Run	Method	Description
SF1_SPA	Default	Using redirect pages and within document query expansion
SF2_SPA	No_Doc_Exp	Using redirect pages
SF3_SPA	SPA_Th2	As run SF1, but allowing lower thresholds
SF4_SPA	No_Redirect	Using within document query expansion
SF5_SPA	SPA_Th2 + No_Doc_Exp	As run SF2, but allowing lower thresholds
KB1_SPA	Emb_Link + SPA_Th2	Using embedding linker and allowing lower thresholds
KB2_SPA	Emb_Link	Using embedding linker
KB3_SPA	Alias_Link	Using alias linker
KB4_SPA	KB1, No_Inv_Check	As run KB1, but turning off inverse check
KB5_SPA	Alias_Link + SPA_Th2	Using alias linker and allowing lower thresholds

LDC max	hop0			hop1			All		
Run	**Prec**	**Rec**	**F1**	**Prec**	**Rec**	**F1**	**Prec**	**Rec**	**F1**
SF1_SPA	0.1327	0.2691	0.1777	0.0000	0.0000	0.0000	0.1327	0.1787	0.1523
SF2_SPA	0.1323	0.2369	0.1698	0.0000	0.0000	0.0000	0.1323	0.1573	0.1437
SF3_SPA	0.0419	**0.3333**	0.0745	0.0000	0.0000	0.0000	0.0419	**0.2213**	0.0705
SF4_SPA	**0.1387**	0.2851	**0.1866**	0.0000	0.0000	0.0000	**0.1387**	0.1893	**0.1601**
SF5_SPA	0.0427	0.3213	0.0754	0.0000	0.0000	0.0000	0.0427	0.2133	0.0712
KB1_SPA	0.2698	0.0683	0.1090	0.0000	0.0000	0.0000	0.1288	0.0453	0.0671
KB2_SPA	0.2743	**0.1245**	**0.1713**	0.0409	**0.0556**	0.0471	0.1338	**0.1013**	0.1153
KB3_SPA	0.2804	0.1205	0.1685	**0.0600**	0.0476	**0.0531**	0.1739	0.0960	**0.1237**
KB4_SPA	0.2742	0.0683	0.1093	0.0000	0.0000	0.0000	0.1298	0.0453	0.0672
KB5_SPA	**0.2963**	0.0643	0.1056	0.0000	0.0000	0.0000	**0.2192**	0.0427	0.0714

Table 17. TACKBP 2016 Cross Language SF & KB Results

Run	Method	Description
SF1_XLING	Eng1 + Spa1	Doc_Exp + Redirect
SF2_XLING	Eng2 + Spa2	Redirect
SF3_XLING	Eng1 + Spa3	Doc_Exp + Redirect + SPA_Th2 (SPA)
SF4_XLING	Eng2 + Spa1	Doc_Exp (SPA) + Redirect
SF5_XLING	Eng4 + Spa2	Doc_Exp (ENG) + Redirect + No_SES (ENG)
KB1_XLING	Eng1 + Spa1	SPA_Th2 (SPA)
KB2_XLING	Eng4 + Spa2	No_DS (ENG)
KB3_XLING	Eng1 (ENG_Th2) + Spa2	ENG_Th2 (ENG)
KB4_XLING	Only USchema	No_LSTM (ENG) + No_LSTM (SPA)
KB5_XLING	Only LSTM	No_USchema (ENG) + No_USchema (SPA)

LDC max	hop0			hop1			All		
Run	**Prec**	**Rec**	**F1**	**Prec**	**Rec**	**F1**	**Prec**	**Rec**	**F1**
SF1_XLING	0.1829	0.1555	**0.1681**	**0.2857**	0.0066	0.0128	0.1843	0.1056	**0.1342**
SF2_XLING	0.1876	0.1423	0.1618	0.1304	**0.0098**	**0.0183**	0.1848	0.0979	0.1280
SF3_XLING	0.0786	**0.1803**	0.1095	0.2000	0.0066	0.0127	0.0795	**0.1220**	0.0963
SF4_XLING	0.1750	0.1505	0.1618	0.1304	**0.0098**	**0.0183**	0.1731	0.1034	0.1294
SF5_XLING	**0.1946**	0.1323	0.1576	**0.2857**	0.0066	0.0128	**0.1962**	0.0902	0.1235
KB1_XLING	0.2958	0.0935	0.1420	0.0510	0.0377	0.0434	0.1633	0.0748	0.1026
KB2_XLING	0.2658	0.0835	0.1271	0.0708	0.0410	0.0519	0.1719	0.0693	0.0987
KB3_XLING	0.2753	**0.1125**	**0.1597**	0.0723	**0.0557**	**0.0630**	0.1763	**0.0935**	**0.1222**
KB4_XLING	**0.5278**	0.0314	0.0593	**0.1667**	0.0066	0.0126	**0.4375**	0.0231	0.0439
KB5_XLING	0.3457	0.0769	0.1258	0.0874	0.0295	0.0441	0.2337	0.0610	0.0968

To simplify the descriptions, we use the same abbreviations in the method column of Table 15 and Table 16. The parenthesis indicates the setting is applied to which language.

4.6 Dynamic Feature Selection Experiments on Sequence Tagging

We evaluate the *Dynamic Feature Selection* [7] work on a number of sequence labeling tasks, including (but not limited to) named entity recognition. We evaluate both the speed and accuracy of the Dynamic Feature Selection method on part of speech (POS) tagging, dependency parsing, and named entity recognition (NER).

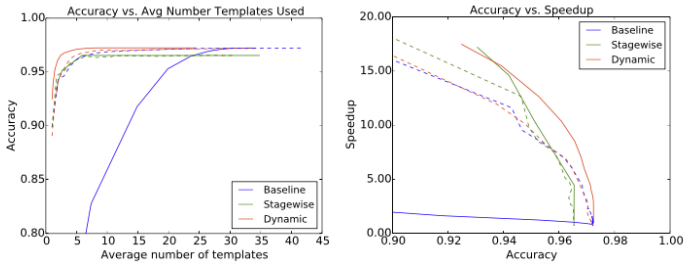
All experiments were performed on the same 2010 Opteron machine with a 2.1GHz AMD processor. We compare for systems: a *baseline* approach which is a linear model with comparable accuracy to state of the art methods at the time of publication, a *stagewise* approach in which the model learns parameters of the templates one at a time, fixing the model parameters of each template as training goes along, a *fixed* approach, which uses the prefix of the feature templates, and the *dynamic* approach with different settings of the confidence margin parameter. We begin with evaluating part of speech tagging, where speed is measured by the number of tokens per second and accuracy is measured by accuracy. The Penn Treebank Wall Street Journal corpus is used for evaluation with the standard train/dev/test split [56]. The baseline tagger has a speed of approximately 23,000 tokens per second, which is approximately the same as the greedy Stanford CoreNLP left3words POS tagger. Significantly higher absolute speeds for all methods can be attained on more modern machines. We increase the speed of our baseline POS tagger by a factor of 5.2x without falling below 97% test accuracy. By tuning our training method to more aggressively prune templates, we achieve speedups of over 10x while providing accuracy higher than 96%. It is worth noting that the results for our method (dynamic) are all obtained from a single trained model (with hyperparameters optimized for m = 50, which we observed gave a good speedup with nearly no loss in accuracy on the development set), the only difference being that we varied the margin at test time. Superior results for m = 50 could likely be obtained by optimizing hyperparameters for the desired margin. The dynamic method selects often only a small fraction of the templates. The majority of test tokens can be tagged using only the first few templates: just over 40% use one template, and 75% require at most four templates, while maintaining 97.17% accuracy. On average 6.71 out of 46 templates are used, though a small set of complicated instances never surpass the margin and use all 46 templates.

Table 18. Dynamic Feature Selection POS tagging results

Model/m	Tok.	Unk.	Feat.	Speed
Baseline	97.22	88.63	46	1x
Stagewise	96.54	83.63	9.50	2.74
Fixed	89.88	56.25	1	16.16x
Fixed	94.66	60.59	3	9.54x
Fixed	96.16	87.09	5	7.02x
Fixed	96.88	88.81	10	3.82x
Dynamic/15	96.09	83.12	1.92	**10.36x**
Dynamic/35	97.02	88.26	4.33	**5.22x**
Dynamic/45	97.16	88.84	5.87	3.97x
Dynamic/50	**97.21**	88.95	6.89	3.41x

Comparison of our models using different margins m, with speeds measured relative to the baseline.

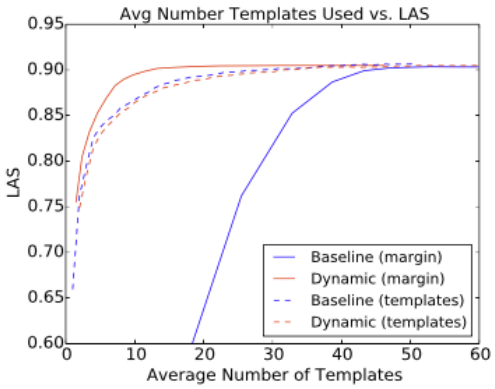
Table 19. Dynamic Feature Selection POS tagging Accuracy vs Num Templates/Speed



The left-hand plot depicts test accuracy as a function of the average number of templates used to predict. The right hand plot shows speedup as a function of accuracy. Dynamic consistently achieves higher accuracy while using fewer templates in the best ratio of speed to accuracy.

Next, we evaluate the method on greedy non-projective transition based dependency parsing with the baseline model [57]. Our model uses a total of 60 feature templates based mainly on the word form, POS tag, lemma and assigned head label of current and previous input and stack tokens, and parses about 300 sentences/second. Accuracy and speed evaluation in this case are done by Label Attachment Score (LAS) and sentences per second respectively. We again see effective speedups with only small loss in accuracy. Our experimental setup is the same as for part- of-speech tagging. We compare our model (dynamic) to both a single baseline model trained on all features, and a set of 60 models each trained on a prefix of feature templates. Our experiments vary the margin used during prediction (solid) as well as the number of templates used (dashed). As in part-of-speech tagging, we observe significant test-time speedups when applying our method of dynamic feature selection to dependency parsing. With a loss of only 0.04 labeled attachment score (LAS), our model produces parses 2.7 times faster than the baseline. As listed in Table 20, with a more aggressive margin our model can parse more than 3 times faster while remaining above 90% LAS, and more than 5 times faster while maintaining accuracy above 88.5%.

Table 20. Dynamic Feature Selection Dependency Parsing Results



Model/m	LAS	UAS	Feat.	Speed
Baseline	90.31	91.83	60	1x
Fixed	65.99	70.78	1	27.5x
Fixed	86.87	88.81	10	5.51x
Fixed	88.76	90.51	20	2.83x
Fixed	89.04	90.71	30	1.87x
Dynamic/6.5	88.63	90.36	7.81	5.16x
Dynamic/7.1	89.07	90.73	8.57	4.66x
Dynamic/10	90.16	91.70	13.27	3.17x
Dynamic/11	90.27	91.80	15.83	2.71x

The left hand graph shows the parsing speedup as a function of accuracy. The right hand table gives results for each method on the WSJ dataset.

Finally, we evaluate the named entity recognition performance using a baseline model of a greedy left-to-right tagger based on features from Ratinov and Roth's system [40] using a total of 46 feature templates, including surface features such as lemma and capitalization, gazetteer look-ups, and each token's extended prediction history. Precision/Recall/F1 measures on the predicted

mention spans is used as is typically done for NER. Training, tuning, and evaluation are performed on the CoNLL 2003 English data set with the BILOU encoding to denote label spans.

Our baseline model achieves F1 scores of 88.35 and 93.37 on the test and development sets, respectively, and tags at a rate of approximately 5300 tokens per second on the hardware described in the experiments above. We achieve a 2.3x speedup while maintaining F1 score above 88 on the test set.

Table 21. Dynamic Feature Selection NER Results

Model/m	Test F1	Feat.	Speed
Baseline	88.35	46	1x
Fixed	65.05	1	19.08x
Fixed	85.00	10	2.14x
Fixed	85.81	13	1.87x
Dynamic/3.0	87.62	7.23	2.59x
Dynamic/4.0	88.20	9.45	2.32x
Dynamic/5.0	88.23	12.96	1.96x

4.7 Lexicon Infused Phrase Embedding for NER Experiments

Our phrase embeddings are learned on the combination of English Wikipedia and the RCV1 Corpus [58]. Wikipedia contains 8M articles, and RCV1 contains 946K. In addition to phrases discovered from the aforementioned bigram/trigram procedure we add page titles from the English Wikipedia to the list of candidate phrases, as well as all word types. We get a total of about 10M phrases. We restrict the vocabulary to the most frequent 1M phrases. All our reported experiments are on 50-dimensional embeddings. Longer embeddings, while performing better on the semantic similarity task, as seen in Mikolov et al [38] [39]), did not perform as well on NER.

To train phrase embeddings, we use a context of length 21. We use lexicons derived from Wikipedia categories and data from the US Census, totaling K = 22 lexicon classes. We use a randomly selected 0.01% of negative training examples for lexicons. We compare the system with baseline features (no lexicons nor embeddings) to the system with consider phrase embeddings (called “Skip-gram”), lexicon-infused embeddings (called “LexEmb”), Brown clusters (“Brown”), and the lexicons (called “Gaz”) as features.

We applied our models on CoNLL 2003 NER data set. All hyperparameters were tuned by training on the training set, and evaluated on the development set. Then the best hyperparameter values were trained on the combination of training and development data and applied on the test set, to obtain the final results. Table 22 shows the phrase F1 scores of all systems we implemented, as well as state-of-the- art results from the literature. Note that using traditional unsupervised Skip-gram embeddings is worse than Brown clusters. In contrast, our lexicon-infused phrase embeddings Lex-0.01 achieves 90.90—a state-of-the-art F1 score for the test set. This result matches the highest F1 previously reported [59], but does so without using massive private data. Our result is significantly better than the previous best using public data.

Table 22. Lexicon Infused Embedding CoNLL 2003 NER Results

System	Dev	Test
Baseline	92.22	87.93
Baseline + Brown	93.39	90.05
Baseline + Skip-gram	93.68	89.68
Baseline + LexEmb	93.81	89.56
Baseline + Gaz	93.69	89.27
Baseline + Gaz + Brown	93.88	90.67
Baseline + Gaz + Skip-gram	94.23	90.33
Baseline + Gaz + LexEmb	**94.46**	**90.90**
Ando and Zhang (2005)	93.15	89.31
Suzuki and Isozaki (2008)	**94.48**	89.92
Ratinov and Roth (2009)	93.50	90.57
Lin and Wu (2009)	-	**90.90**

Lin and Wu (2009) is the aforementioned system with massive private industrial query-log data in training.

We also evaluate the models on the Ontonotes 5.0 NER dataset. We use the same experimental setup as in the CoNLL 2003 experiment. We observe a similar trend in results on this dataset.

Table 23. Lexicon Infused Embedding Ontonotes 5.0 NER Results

System	Dev	Test
Baseline	79.04	79.85
Baseline + Brown	79.95	81.38
Baseline + Skip-gram	80.59	81.91
Baseline + LexEmbd	80.65	81.82
Baseline + Gaz	79.85	81.31
Baseline + Gaz + Brown	80.53	82.05
Baseline + Gaz + Skip-gram	80.70	**82.30**
Baseline + Gaz + LexEmb	**80.81**	82.24

4.8 Fast and Accurate Entity Recognition with ID-CNNs Experiments

As in the previous sections, we evaluate the Iterated Dilated Convolutional Neural Network (ID-CNN) based named entity recognition model [8] on the CoNLL-2003 and Ontonotes 5.0 datasets. We compare our ID-CNN against strong LSTM and CNN baselines: a Bi-LSTM with local decoding, and one with CRF decoding (Bi-LSTM- CRF). We also compare against a non-dilated CNN architecture with the same number of convolutional layers as our dilated network (4-layer CNN) and one with enough layers to incorporate an effective input width of the same size as that of the dilated network (5-layer CNN) to demonstrate that the dilated convolutions more effectively aggregate contextual information than simple convolutions (i.e. using fewer parameters). We also compare our document-level ID-CNNs to a baseline, which does not share parameters between blocks (called “noshare”) and one that computes loss only at the last block, rather than after every iterated block of dilated convolutions (called “1-loss”).

On CoNLL-2003 English NER, ID-CNN performs on par with a Bi-LSTM not only when used to produce per-token logits for structured inference. The ID-CNN with greedy decoding also

performs on-par with the Bi-LSTM-CRF while running at more than 14 times the speed. We also observe a performance boost in almost all models when broadening the context to incorporate entire documents, achieving an average F1 of 90.65 on CoNLL-2003, out-performing the sentence-level model while still decoding at nearly 8 times the speed of the Bi-LSTM-CRF as seen in Tables 24 and 25.

The results indicate that our ID-CNN is not only a better token encoder than the Bi-LSTM but it is also faster. Table 25 lists relative decoding times on the CoNLL development set, compared to the Bi-LSTM-CRF. We report decoding times using the fastest batch size for each method. The ID-CNN model decodes nearly 50% faster than the Bi-LSTM. With Viterbi decoding, the gap closes somewhat but the ID-CNN-CRF still comes out ahead, about 30% faster than the Bi-LSTM-CRF. The most vast speed improvements come when comparing the greedy ID-CNN to the Bi-LSTM-CRF – our ID-CNN is more than 14 times faster than the Bi-LSTM-CRF at test time, with comparable accuracy.

In Table 26 we show that adding document-level context improves every model on CoNLL-2003. Incorporating document-level context further improves our greedy ID-CNN model, attaining 90.65 average F1. We believe this model sees greater improvement with the addition of document-level context than the Bi-LSTM-CRF due to the ID-CNN learning a feature function better suited for representing broad context, in contrast with the Bi-LSTM which, though better than a simple RNN at encoding long memories of sequences, may reach its limit when provided with sequences more than 1,000 tokens long such as entire documents.

Table 24. ID-CNN on CoNLL 2003 NER using Sentence Level Context (F1)

Model	F1
Ratinov and Roth (2009)	86.82
Collobert et al. (2011)	86.96
Lample et al. (2016)	90.33
Bi-LSTM	89.34 ± 0.28
4-layer CNN	89.97 ± 0.20
5-layer CNN	90.23 ± 0.16
ID-CNN	90.32 ± 0.26
Collobert et al. (2011)	88.67
Passos et al. (2014)	90.05
Lample et al. (2016)	90.20
Bi-LSTM-CRF (re-impl)	90.43 ± 0.12
ID-CNN-CRF	**90.54 ± 0.18**

No models use character embeddings or lexicons.
Top models (those that sit above double bar) are greedy, bottom models use Viterbi inference.

Table 25. ID-CNN on CoNLL 2003 NER using Sentence Level Context (Speed)

Model	Speed
Bi-LSTM-CRF	1×
Bi-LSTM	9.92×
ID-CNN-CRF	1.28×
5-layer CNN	12.38×
ID-CNN	14.10×

Table 26. ID-CNN on CoNLL 2003 NER using Document Level Context (F1)

Model	F1
4-layer ID-CNN (sent)	90.32 ± 0.26
Bi-LSTM-CRF (sent)	90.43 ± 0.12
4-layer CNN × 3	90.32 ± 0.32
5-layer CNN × 3	90.45 ± 0.21
Bi-LSTM	89.09 ± 0.19
Bi-LSTM-CRF	90.60 ± 0.19
ID-CNN	**90.65** ± **0.15**

Table 27. ID-CNN on CoNLL 2003 NER using Document Level Context (Speed)

Model	Speed
Bi-LSTM-CRF	1×
Bi-LSTM	4.60×
ID-CNN	7.96×

Fastest batch per model is reported.

We observe similar result trends on OntoNotes 5.0 as we do on CoNLL 2003. Table 28 lists overall F1 scores of our models compared to those in the existing literature. The greedy Bi-LSTM outperforms the lexicalized greedy model of Ratinov and Roth [40], and our ID-CNN outperforms the Bi-LSTM as well as the more complex model of Durrett and Klein [60] which leverages the parallel co-reference annotation available in the OntoNotes corpus to predict named entities jointly with entity linking and co-reference. Our greedy model is outperformed by the Bi-LSTM-CRF reported in Chiu and Nichols [61] as well as our own re-implementation, which appeared to be the new state-of-the-art on this dataset at the time of publication. The gap between our greedy model and those using Viterbi decoding is wider than on CoNLL, perhaps because of the diverse set of entities in OntoNotes, which also tend to be much longer. Incorporating greater context significantly boosts the score of our greedy model on OntoNotes, whereas the Bi-LSTM-CRF performs more poorly.

Table 28. ID-CNN on Ontonotes 5.0 NER

Model	F1	Speed
Ratinov and Roth (2009)[6]	83.45	
Durrett and Klein (2014)	84.04	
Chiu and Nichols (2016)	86.19 ± 0.25	
Bi-LSTM-CRF	86.99 ± 0.22	1×
Bi-LSTM-CRF-Doc	86.81 ± 0.18	1.32×
Bi-LSTM	83.76 ± 0.10	24.44×
ID-CNN-CRF (1 block)	86.84 ± 0.19	1.83×
ID-CNN-Doc (3 blocks)	85.76 ± 0.13	21.19×
ID-CNN (3 blocks)	85.27 ± 0.24	13.21×
ID-CNN (1 block)	84.28 ± 0.10	26.01×

4.9 Within Document Coreference Experiments

We evaluate our model [45] described in Section 3.2.4 on the ACE 2004 annotated dataset [62]. We split the corpus into training, development, and test sets, resulting in 268 documents in the train set, 107 documents in the test set, and 68 documents in the development set [63]. The data is processed using standard open source tools to segment the sentences and tokenize the corpus, and using the OpenNLP2 tagger to obtain the POS tags.

To provide the initial ranked list of entity candidates from Wikipedia, we query the KB Bridge system (Dalton and Dietz, 2013) with the proper name mentions. KB Bridge is an information-retrieval-based entity linking system that connects the query mentions to Wikipedia entities using a sequential dependence model.

We consider the following approaches:

- Wikipedia Linking [64]: Use KB Bridge to link mentions to Wikipedia (determining within document coreference links based on these resolutions). The non-pronoun mentions are linked to these proper nouns if the mention string matches any of the entity titles or anchor texts.
- Bengston and Roth (2008) [63]: A pairwise coreference model containing a rich set of features, as described and evaluated in Bengston and Roth (2008).
- Baseline: Our implementation of a pairwise model that is similar to the approach in Bengston and Roth (2008) [63] with minor alterations. This slightly outperforms Bengston and Roth (2008).
- Dynamic linking: The proposed approach which re-ranks the list of candidate entity links for a mention during inference.
- Static linking: The same as the proposed approach without merging candidate lists during inference.

Incorporating Wikipedia and anchor text information from the web with a fixed alignment (*static linking*) further improves performance by 0.54 B3 F1 points. Using *dynamic linking*, which improves the alignment during inference, achieves another 0.55 F1 point improvement, which is 1.09 F1 above our baseline, 1.41 F1 above the best pairwise classification system at the time of publication (corresponding to an error reduction of 7.4%), and 0.4 F1 above the current state-of-

art on this dataset [47]. The improvement of the dynamic linking approach over our baselines is consistent across the various evaluation metrics.

Table 29. ACE 2004 Within Document Coreference

Method	Pairwise		MUC		CEAF		B^3	
	P / R	F1	P / R	F1	P / R	F1	P / R	F1
Culotta et al. (2007)	-	-	-	-	-	-	86.7 73.2	79.3
Raghunathan et al. (2010)	71.6 46.2	56.1	80.4 71.8	75.8	-	-	86.3 75.4	80.4
Stoyanov and Eisner (2012)	-	-	-	**80.1**	-	-	-	81.8
Wiki-linking	64.15 14.99	24.30	74.41 28.39	41.10	58.54 58.4	58.47	92.89 57.21	70.81
Bengston and Roth (2008)	-	-	82.7 69.9	75.8	-	-	88.3 74.5	80.8
Baseline	66.56 47.07	55.14	82.84 72.02	77.05	75.58 75.40	75.49	87.02 75.97	81.12
Static Linking	82.53 40.80	54.61	88.39 66.93	76.18	75.33 75.35	75.44	93.10 72.72	81.66
Dynamic Linking	72.20 47.40	**57.23**	85.07 72.02	78.01	76.55 76.37	**76.46**	89.37 76.12	**82.21**

4.10 Extreme Clustering Experiments

Entity resolution without a reference knowledge base is a clustering problem. In particular, we expect it to be an *extreme clustering* problem, in which there is not only a massive number of points to cluster but also a massive number of clusters. At the time of this report we are still working on applying PERCH to entity resolution problems. In place of this, we present results evaluating PERCH on number of extreme clustering datasets.

We compare the following algorithms:

- PERCH - The algorithm described in section 3.3.2. We use various modes of the algorithm depending on the size of the problem. For beam search, we use a default width of 5 (per thread, 24 threads). We only run in collapsed mode for ILSVRC12 and ImageNet (100K) where L = 50000
- BIRCH [65] - A top-down hierarchical clustering method where each internal node represents its leaves via mean and variance statistics. Points are inserted greedily using the node statistics when the branching factor is exceeded, a powerful split operation is invoked. Importantly no rotations are performed.
- HAC- Hierarchical agglomerative clustering (various linkages). The algorithm repeatedly merges the two subtrees that are closest according to some measure to form a larger subtree.
- Mini-batch HAC (MB-HAC) - HAC (various linkages) made to run online with mini-batching. The algorithm maintains a buffer of subtrees and must merge two subtrees in the buffer before observing the next point. We use buffers of size 2K and 5K and centroid (cent.) and complete (com.) linkages
- K-means - Lloyd's algorithm, which alternates between assigning points to centers and recomputing centers based on the assignment. We use the K-means++ initialization (Arthur et al, 2007).
- Stream K-means++ (SKM++) [66]- A streaming algorithm that computes a representative subsample of the points (a coreset) in one pass, runs K-means++ on the subsample, and in a second pass assigns points greedily to the nearest cluster center.
- Mini-batch K-means (MB-KM) [67] - An algorithm that optimizes the K-means objective function via mini-batch stochastic gradient descent. The implementation we use includes many heuristics, including several initial passes through the data to initialize centers via K-means++, random restarts to avoid local optima, and early stopping (Pedregosa et al, 2011).

In robustness experiments, we also study a fully online version of this algorithm, with random initialization and without early stopping or reassignments.

- BICO [68] - An algorithm for optimizing the K-means objective that creates coresets via a streaming approach using a BIRCH- like data structure. K-means++ is run on the coresets and then points are assigned to the inferred centers.
- DBSCAN [69] - A density based method that computes nearest-neighbor balls around each point, merges overlapping balls, and builds a clustering from the resulting connected components.
- Hierarchical K-means(HKMeans)-top-down, divisive, hierarchical clustering. At each level of the hierarchy, the remaining points are split into two groups using K-means.

Since few benchmark clustering datasets exhibit a large number of clusters, we conduct experiments with large-scale classification datasets that naturally contain many classes and simply omit the class labels. Although smaller datasets are not our primary focus, we also measure the performance of PERCH on standard clustering benchmarks. The datasets used are:

- ALOI - (Amsterdam Library of Object Images [70]) contains images and is used as an extreme classification benchmark.
- Speaker - The NIST I-Vector Machine Learning Challenge speaker detection dataset [71] The goal is to cluster recordings from the same speaker together. We cluster the whitened development set (scripts provided by the challenge).
- ILSVRC12 - The ImageNet [72] Large Scale Visual Recognition Challenge 2012 [73]. The class labels are used to produce a ground truth clustering. We generate representations of each image from the last layer of the Inception neural network [74].
- ILSVRC12 (50K) - a 50K subset of ILSVRC12.
- ImageNet (100K) - a 100K subset of the ImageNet database. Classes are sampled proportional to their frequency in the database.
- Covertype - forest cover types (benchmark).
- Glass - different types of glass (benchmark).
- Spambase-email data of spam and not-spam(benchmark).
- Digits- a subset of a hand written digits dataset(benchmark).

The Covertype, Glass, Spambase and Digits datasets are provided by the UCI Machine Learning Repository [75].

Table 30. Extreme Clustering Dataset Statistics

	Name	Clusters	Points	Dim.
Large Data sets	ImageNet (100K)	17K	100K	2048
	Speaker	4958	36,572	6388
	ILSVRC12	1000	1.3M	2048
	ALOI	1000	108K	128
	ILSVRC12 (50K)	1000	50K	2048
	CoverType	7	581K	54
Small Benchmarks	Digits	10	200	64
	Glass	6	214	10
	Spambase	2	4601	57

PERCH and many of the other baselines build cluster trees, and, as such, they can be evaluated using dendrogram purity. In Table 31, we report the dendrogram purity, averaged over random shufflings of each dataset, for the 6 large datasets, and for the hierarchical clustering algorithms (PERCH, BIRCH, MB-HAC variants, HKMeans and HAC variants). Bold indicates the best performing scalable approach; italic indicates the best performing approach overall. The top 5 rows in the table correspond to incremental algorithms, while the bottom 4 are batch algorithms (with the two below the horizontal bar being unscalable to large N). The comparison demonstrates the quality of the incremental algorithms, as well as the degree of scalability of the batch methods. Unsurprisingly, we were not able to run some of the batch algorithms on the larger datasets. PERCH consistently produces trees with highest dendrogram purity amongst all incremental methods. PERCH-B, which uses approximate nearest-neighbor search, is worse, but still consistently better than the baselines.

Table 31. Extreme Clustering - Hierarchical Clustering / Dendrogram Purity Results

Method	CovType	ILSVRC12 (50k)	ALOI	ILSVRC 12	Speaker	ImageNet (100k)
PERCH	***0.45 ± 0.004***	**0.53 ± 0.003**	***0.44 ± 0.004***	–	**0.37 ± 0.002**	***0.07 ± 0.00***
PERCH-BC	***0.45 ± 0.004***	0.36 ± 0.005	0.37 ± 0.008	***0.21 ± 0.017***	0.09 ± 0.001	0.03 ± 0.00
BIRCH (incremental)	0.44 ± 0.002	0.09 ± 0.006	0.21 ± 0.004	0.11 ± 0.006	0.02 ± 0.002	0.02 ± 0.00
MB-HAC-Com.	–	0.43 ± 0.005	0.15 ± 0.003	–	0.01 ± 0.002	–
MB-HAC-Cent.	0.44 ± 0.005	0.02 ± 0.000	0.30 ± 0.002	–	–	–
HKMmeans	0.44 ± 0.001	0.12 ± 0.002	***0.44 ± 0.001***	0.11 ± 0.003	0.12 ± 0.002	0.02 ± 0.00
BIRCH (rebuild)	0.44 ± 0.002	0.26 ± 0.003	0.32 ± 0.002	–	0.22 ± 0.006	0.03 ± 0.00
HAC-Avg	–	*0.54*	–	–	*0.55*	–
HAC-Complete	–	0.40	–	–	0.40	–

We also compare PERCH to the flat-clustering algorithms described above. Here we evaluate a K-way clustering via the Pairwise F1 score, which given ground truth clustering C and estimate C' is the harmonic mean of the precision and recall between the pairs of points that are clustered together in C and C'. In this section, we compare PERCH to MB-KM, SKM++, BICO, DBSCAN, K-means (with K-means++ initialization) and HAC where the tree construction terminates once K-subtrees remain. All algorithms, including PERCH, use the true number of clusters as an input parameter (except DBSCAN, which does not take the number of clusters as input).

To select a flat clustering from PERCH's cluster tree, we use the following greedy heuristic. While the tree *T* does not have *K* leaves, collapse the internal node *v* with the smallest *cost.* We define the *cost* of a node *v, cost(v)* as the maximum length diagonal of *v*'s bounding box multiplied by *|lvs(v)|*. *Cost(v)* can be thought of as an upper bound on the K-means cost of *v*. We also experimented with other heuristics, but found this one to be quite effective. The results of the experiment are in Table 32. Again, bold indicates the best performing scalable approach; italic indicates the best performing approach overall. As in the case of dendrogram purity, PERC competes with or outperforms all the scalable algorithms (i.e., above the horizontal line) on all datasets, even though we use a naive heuristic to identify the final flat clustering from the tree. K-means, which could not be run on the larger datasets, is able to achieve a clustering with 60.4 F1 on ILSVRC (50K) and 32.19 F1 on Speaker; HAC with average-linkage and HAC with complete-linkage achieve scores of 40.26 and 44.3 F1 respectively on Speaker. This is unsurprising because in each iteration of both K-means and HAC, the algorithm has access to the entire dataset. We were able to run K-means on the Covertype dataset and achieve a score of 24.42 F1. We observe that K-means is able to converge quickly, which is likely due to the K-means++ initialization and

the small number of true clusters (7) in the dataset. Since PERCH performs well on each of the datasets in terms of dendrogram purity, it may be possible to extract better flat clusterings from the trees it builds with a different pruning heuristic.

Table 32. Extreme Clustering - Flat Clustering / Pairwise F1 Results

Method	CoverType	ILSVRC 12 (50k)	ALOI	ILSVRC 12	Speaker	ImageNet (100K)
PERCH	22.96 ± 0.7	**54.30 ± 0.3**	***44.21 ± 0.2***	–	**31.80 ± 0.1**	***6.178 ± 0.0***
PERCH-BC	22.97 ± 0.8	37.98 ± 0.5	37.48 ± 0.7	25.75 ± 1.7	1.05 ± 0.1	4.144 ± 0.04
SKM++	23.80 ± 0.4	28.46 ± 2.2	37.53 ± 1.0	–	–	–
BICO	***24.53 ± 0.4***	45.18 ± 1.0	32.984 ± 3.4	–	–	–
MB-KM	24.27 ± 0.6	51.73 ± 1.8	40.84 ± 0.5	***56.17 ± 0.4***	1.73 ± 0.141	5.642 ± 0.00
DBSCAN	–	16.95	–	–	22.63	–
Kmeans	24.42 ± 0.00	*60.40 ± 0.5*	39.311 ± 0.3	–	32.185 ± 0.01	–
HAC-Avg	–	–	–	–	40.258	–
HAC-Complete	–	18.28	–	–	*44.297*	–

We compare the best performing methods above in terms of running time and accuracy. We focus on PERCH-BC, BIRCH, and HKMeans. Each of these algorithms has various parameter settings that typically govern a trade-off between accuracy and running time, and in our experiment, we vary these parameters to better understand this trade-off. The specific parameters are:

- PERCH BC: beam-width,collapse threshold,
- BIRCH:branching factor,
- HKMeans: number of iterations per level.

In Figure 15, we plot the dendrogram purity as a function of running time for the algorithms as we vary the relevant parameter. We use the ILSVRC12 (50K) dataset and run all algorithms on the same machine (28 cores with 2.40GHz processor). The results show that PERCH - BC achieves a better trade-off between dendrogram purity and running time than the other methods. Except for in the uninteresting low purity regime, our algorithm achieves the best dendrogram purity for a given running time. HKMeans with few iterations can be faster, but it produces poor clusterings, and with more iterations the running time scales quite poorly. BIRCH with rebuilding performs well, but PERCH- BC performs better in less time.

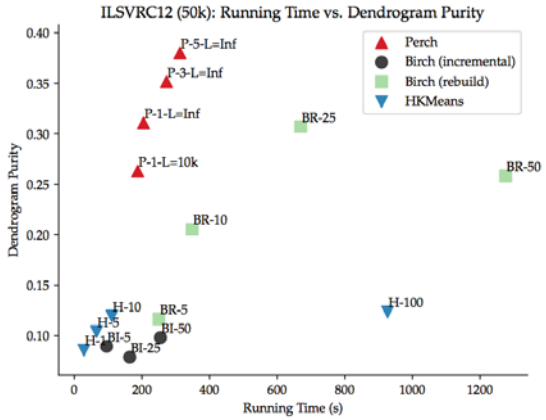


Figure 15. Extreme Clustering - Speed vs. Dendrogram Purity

4.11 Multi-Sense Word Embedding Experiments

We evaluate the representations learned using our Multi-Sense Skip-gram model [11] on two word similarity tasks: the WordSim-353 [76] dataset and the Contextual Word Similarities (SCWS) dataset [77]. Following Huang et al [77], we train our models on the April 2010 snapshot of the Wikipedia corpus [78]. In all our experiments, we remove all the words with less than 20 occurrences and use a maximum context window (N) of length 5 (5 words before and after the word occurrence). We fix the number of senses (K) to be 3 for the MSSG model unless otherwise specified. Our hyperparameter values were selected by a small amount of manual exploration on a validation set. We set the facility location cost to be -0.5.

WordSim-353 is a standard dataset for evaluating word vector representations. It consists of a list of pairs of word types, the similarity of which is rated in an integral scale from 1 to 10. Pairs include both monosemic and polysemic words. These scores to each word pairs are given without any contextual information, which makes them tricky to interpret.

To overcome this issue, Stanford's Contextual Word Similarities (SCWS) dataset was developed by Huang et al (2012). The dataset consists of 2003 word pairs and their sentential contexts. It consists of 1328 noun-noun pairs, 399 verb-verb pairs, 140 verb-noun, 97 adjective-adjective, 30 noun-adjective, 9 verb-adjective, and 241 same-word pairs.

We use four similarity functions to make predictions from the multiple sense word embeddings: avgSim, which computes the average similarity over all embeddings for each word, ignoring information from the context, *avgSimC*, which weighs the similarity between each pair of senses by how well does each sense fit the context at hand, *globalSim*, which uses each word's global context vector, ignoring the many senses, and *localSim*, which selects a single sense for each word based independently on its context and selects the similarity based on the max similarity.

We report the Spearman correlation (x100) between a model's similarity scores and the human judgments in the datasets.

On WordSim-353 task, we see that our model performs significantly better than the previous neural network model for learning multi-representations per word [77]. Among the methods that learn low-dimensional and dense representations, our model performs slightly better than Skip-gram.

Table 33. Multi-Sense Skip-Gram Model WordSim-353 Results

Model	$\rho \times 100$
HLBL	33.2
C&W	55.3
Skip-gram-300d	70.4
Huang et al-G	22.8
Huang et al-M	64.2
MSSG 50d-G	60.6
MSSG 50d-M	63.2
MSSG 300d-G	69.2
MSSG 300d-M	**70.9**
NP-MSSG 50d-G	61.5
NP-MSSG 50d-M	62.4
NP-MSSG 300d-G	69.1
NP-MSSG 300d-M	68.6
Pruned TF-IDF	73.4
ESA	75
Tiered TF-IDF	76.9

In the SCWS task, when the words are given with their context, our model achieves new state-of-the-art results on SCWS as shown in the Table 34. The previous state-of-art model [77] on this task achieves 65.7% using the avgSimC measure, while the MSSG model achieves the best score of 69.3% on this task.

Table 34. Multi-Sense Skip-Gram Model SCWS Results

Model	globalSim	avgSim	avgSimC	localSim
TF-IDF	26.3	-	-	-
Collobort & Weston-50d	57.0	-	-	-
Skip-gram-50d	63.4	-	-	-
Skip-gram-300d	65.2	-	-	-
Pruned TF-IDF	62.5	60.4	60.5	-
Huang et al-50d	58.6	62.8	65.7	26.1
MSSG-50d	62.1	64.2	66.9	49.17
MSSG-300d	65.3	67.2	**69.3**	57.26
NP-MSSG-50d	62.3	64.0	66.1	50.27
NP-MSSG-300d	**65.5**	**67.3**	69.1	**59.80**

4.12 Word Representations via Gaussian Embeddings Experiments

We evaluate our work on learning Gaussian word representations [12] on several tasks such as including modeling asymmetric and linguistic relationships, uncertainty, and word similarity. All Gaussian experiments are conducted with 50-dimensional vectors, with diagonal variances except where noted otherwise. Unsupervised embeddings are learned on the concatenated ukWaC and WaCkypedia corpora [79], consisting of about 3 billion tokens. This matches the experimental setup used by Baroni et al. [80], aside from leaving out the small British National Corpus, which is not publicly available and contains only 100 million tokens. All word types that appear less than 100 times in the training set are dropped, leaving a vocabulary of approximately 280 thousand word types.

We evaluate quantitatively on the Entailment dataset of Baroni et al. [80]. We compare variances learned jointly during embedding training by using the expected likelihood objective, with empirical variances gathered from contexts on pre-trained word2vec-style embeddings. We compare both diagonal (D) and spherical (S) variances, using both cosine similarity between means, and KL divergence. Baseline asymmetric measurements, such as the difference between the sizes of the two embeddings, did worse than the cosine. We see that KL divergence between the entailed and entailing word does not give good performance for the empirical variances, but beats the count-based balAPinc measure when used with learned variances. We include the best F1 score (by picking the optimal threshold at test) because this is used by Baroni et al. [80], but we believe AP is better to demonstrate the correlation of various asymmetric and symmetric measures with the entailment data.

Table 35. Entailment Experiments with Gaussian Word Representations

Model	Test	Similarity	Best F1	AP
Baroni et al. (2012)	E	balAPinc	**75.1**	–
Empirical (D)	E	KL	70.05	.68
Empirical (D)	E	Cos	76.24	.71
Empirical (S)	E	KL	71.18	.69
Empirical (S)	E	Cos	76.24	.71
Learned (D)	E	KL	79.01	**.80**
Learned (D)	E	Cos	76.99	.73
Learned (S)	E	KL	**79.34**	.78
Learned (S)	E	Cos	77.36	.73

We compare empirical and learned variances, both diagonal (D) and spherical (S). E is the dataset of Baroni et al. [80]. Measures of similarity are symmetric (cosine between means) and asymmetric (KL) divergence for Gaussians. balAPinc is an asymmetric similarity measure specific to sparse, distributional count-based representations.

We also evaluate the embeddings on seven different standard word similarity benchmarks [76][81][82][83][84][85][86]. A comparison to all of the state of the art word-embedding numbers for different dimensionalities as in (Baroni et al., 2014) is out of the scope of this evaluation. However, we note that the overall performance of our 50-dimensional embeddings matches or beats reported numbers on these datasets for the 80-dimensional Skip-Gram vectors at wordvectors.org [87], as well as our own Skip-Gram implementation. Note that the numbers are

not directly comparable since we use a much older version of Wikipedia (circa 2009) in our WaCkypedia dataset, but this should not give us an edge.

Table 36. Word Similarity Experiments with Gaussian Word Representations

Dataset	SG (50d)	SG (100d)	LG/50/m/S	LG/50/d/S	LG/50/m/D	LG/50/d/D
SimLex	29.39	31.13	**32.23**	29.84	**31.25**	30.50
WordSim	59.89	59.33	**65.49**	62.03	**62.12**	61.00
WordSim-S	69.86	70.19	**76.15**	73.92	**74.64**	72.79
WordSim-R	53.03	54.64	**58.96**	54.37	**54.44**	53.36
MEN	70.27	70.70	**71.31**	69.65	**71.30**	70.18
MC	63.96	66.76	**70.41**	69.17	67.01	**68.50**
RG	70.01	69.38	71.00	**74.76**	70.41	**77.00**
YP	39.34	35.76	41.50	**42.55**	36.05	**39.30**
Rel-122	49.14	51.26	**53.74**	51.09	52.28	**53.54**

We evaluate our learned Gaussian embeddings (LG) with spherical (S) and diagonal (D) variances, on several word similarity benchmarks, compared against standard Skip- Gram (SG) embeddings on the trained on the same dataset. We evaluate Gaussian embeddings with both cosine between means (m), and cosine between the distributions themselves (d) as defined by the expected likelihood inner product.

While it is good to sanity-check that our embedding algorithms can achieve standard measures of distributional quality, these experiments also let us compare the different types of variances (spherical and diagonal). We also compare against Skip-Gram embeddings with 100 latent dimensions, since our diagonal variances have 50 extra parameters.

We see that the embeddings with spherical covariances have an overall slight edge over the embeddings with diagonal covariances in this case, in a reversal from the entailment experiments. This could be due to the diagonal variance matrices making the embeddings more axis-aligned, making it harder to learn all the similarities and reducing model capacity. To test this theory, we plotted the absolute values of components of spherical and diagonal variance mean vectors on a q-q plot and noted a significant off-diagonal shift, indicating that diagonal variance embedding mean vectors have “spikier” distributions of components, indicating more axis-alignment.

We also see that the distributions with diagonal variances benefit more from including the variance in the comparison (d) than the spherical variances. Generally, the data sets in which the cosine between distributions (d) outperforms cosine between means (m) are similar for both spherical and diagonal covariances. Using the cosine between distributions never helped when using empirical variances, so we do not include those numbers.

4.13 Chains of Reasoning using Recurrent Neural Networks Experiments

To evaluate our work on reasoning over paths in knowledge bases/graphs, we measure the relation prediction on a dataset of Freebase relations supplemented with textual relations from ClueWeb, which was related by our work [13]. In this evaluation, the system is given a pair of entities and it ranks the relations, which might exist between these two entities. We use the average precision metric to score the ranking of relations for an entity pair. Mean average precision (MAP) is used as an overall metric; it is the mean of the average precision scores across all entity pairs.

The dimension of the relation type representations and the RNN hidden states are d, h = 250 and the entity and type embeddings have m = 50 dimensions. The model in Neelakantan et al [13] has sigmoid units as their activation function. In Das et al [14], we found rectifier units (ReLU) to work much better [88].

In the table below, we see that the LogSumExp pooling function is the most effective. This demonstrates the importance of considering information from all the paths. However, Avg. pooling performs the worst, which shows that it is also important to weigh the paths scores according to their values. We compare our model in Das et al [14], "single model", with two other multi-hop models - Path-RNN [13] and PRA [89]. Both of these approaches train an individual model for each query relation. We also experiment with another extension of PRA that adds bigram features (PRA + Bigram). Additionally, we run an experiment re-placing the max-pooling of Path-RNN with LogSumExp.

We also see that incorporating entity and entity type information is beneficial. We achieve the best performance when we represent entities as a function of their annotated types in Freebase (Single-Model + Types) ($p < 0.005$). In comparison, learning separate representations of entities (Single-Model + Entities) gives slightly worse performance. This is primarily because we encounter many new entities during test time, for which our model does not have a learned representation.

Table 37. Chains of Reasoning Relation Prediction

Model	Performance (%MAP)	Pooling
Single-Model	68.77	Max
Single-Model	55.80	Avg.
Single-Model	68.20	Top(k)
Single-Model	**70.11**	LogSumExp
PRA	64.43	n/a
PRA + Bigram	64.93	n/a
Path-RNN	65.23	Max
Path-RNN	68.43	LogSumExp
Single-Model	**70.11**	LogSumExp
PRA + Types	64.18	n/a
Single-Model	70.11	LogSumExp
Single-Model + Entity	71.74	LogSumExp
Single-Model + Types	**73.26**	LogSumExp
Single-Model + Entity + Types	72.22	LogSumExp

4.14 Neural Programmer Experiments

We apply Neural Programmer on the WikiTableQuestions dataset [90] and compare it to different non-neural baselines including a natural language semantic parser developed by Pasupat & Liang [90].

We use the train, development, and test split given by Pasupat & Liang [90]. The dataset contains 11321, 2831, and 4344 examples for training, development, and testing respectively. We use their tokenization, number and date pre-processing. There are examples with answers that are neither number answers nor phrases selected from the table. We ignore these questions during training but the model is penalized during evaluation following Pasupat & Liang [90]. The tables provided in the test set are unseen at training, hence requiring the model to adapt to unseen column names at

test time. We train only on examples for which the provided table has less than 100 rows since we run out of GPU memory otherwise, but consider all examples at test time.

Table 37 shows the performance of our model in comparison to baselines from Pasupat & Liang [37]. The best result from Neural Programmer is achieved by an ensemble of 15 models. The only difference among these models is that the parameters of each model is initialized with a different random seed. We combine the models by averaging the predicted softmax distributions of the models at every time-step. While it is generally believed that neural network models require a large number of training examples compared to simpler linear models to get good performance, our model achieves competitive performance on this small dataset containing only 10,000 examples with weak supervision.

We did not get better results either by using pre-trained word vectors [38][39] or by pre-training the question RNN with a language modeling objective [91]. A possible explanation is that the word vectors obtained from unsupervised learning may not be suitable to the task under consideration. For example, the learned representations of words like maximum and minimum from unsupervised learning are usually close to each other but for our task it is counter-productive. We consider replacing soft selection with hard selection and training the model with the REINFORCE algorithm [92]. The model fails to learn in this experiment, probably because the model has to search over millions of symbolic programs for every input question making it highly unlikely to find a program that gives a reward. Hence, the parameters of the model are not updated frequently enough.

Table 38. Neural Programmer WikiTableQuestions Results

Method	Dev Accuracy	Test Accuracy
Baselines from Pasupat & Liang (2015)		
Information Retrieval System	13.4	12.7
Simple Semantic Parser	23.6	24.3
Semantic Parser	37.0	37.1
Neural Programmer		
Neural Programmer	34.1	34.2
Ensemble of 15 Neural Programmer models	37.5	37.7
Oracle Score with 15 Neural Programmer models	50.5	-

5.0 CONCLUSION

In this work, we described methods of automatic knowledge base construction and reasoning on knowledge bases based on Universal Schema, now an influential and high-cited approach to knowledge extraction and representation. We motivated these methods as alternatives to fixed schema knowledge base construction and demonstrated their effectiveness at relation and entity type prediction. We presented extensions of these techniques, which generalize to unseen entity pairs (“Row-less” Universal Schema) as well as unseen relations (“Column-less” Universal Schema). We presented experimental results highlighting the generalization ability of these methods and presented results demonstrating the effectiveness using these methods under limited resources in the TAC-KBP in Spanish where the system was ranked 1st in the Spanish track of the competition. We present efficient and effective methods for named entity recognition using

dynamic feature selection and dilated convolutional neural networks. We found these models offered significant speed up with either no or minor loss in accuracy. We presented methods that reason on knowledge base structures themselves to answer queries. These methods allow for complex reasoning using chains of facts in a knowledge graph and introduce a method for inducing programs from questions, advancing our work toward natural language interfaces to structured knowledge bases.

6.0 RECOMMENDATIONS

In this work we demonstrate the effectiveness of Universal Schema vector embedding representations for knowledge base population and reasoning. There is of course, much more work to be done. We believe there is significant future work to be done in: unifying entity types and entity resolution, joint segmentation and entity typing and relations, natural language interfaces to knowledge bases, allowing for and reasoning about human edits to a knowledge base, representing provenance for knowledge base. There are also engineering considerations in building these systems and creating tools for effective browsing/navigation in knowledge bases, especially in presenting users information based on the embedded representations of the facts.

7.0 REFERENCES

[1] Bollacker, K., Evans, C., Paritosh, P., Sturge, T., Taylor, J. "Freebase: a collaboratively created graph database for structuring human knowledge," *SIGMOD*, (2008)

[2] Riedel, S., Yao, L., Marlin, B., McCallum, A. "Relation Extraction with Matrix Factorization and Universal Schemas." *NAACL*, (2013).

[3] Yao, L., Riedel, S., Marlin, B., McCallum, A., "Universal Schema for Entity Type Prediction," *Workshop on Automated Knowledge Base Construction (AKBC)*, (2013).

[4] Verga, P., Belanger, D., Strubell, E., Roth, B., McCallum, A., "Multilingual Relation Extraction using Compositional Universal Schema," *NAACL*, (2016).

[5] Verga, P., Neelakantan, A., McCallum, A., "Generalizing to Unseen Entities and Entity Pairs with Row-less Universal Schema," *EACL*, (2017).

[6] Passos, A., Kumar, V., McCallum, A., "Lexicon Infused Phrase Embeddings for Named Entity Resolution," *CoNLL*, (2014).

[7] Strubell, E. Vilnis, L., Silverstein, K., McCallum, A., "Learning Dynamic Feature Selection for Fast Sequential Prediction," *ACL*, (2015).

[8] Strubell, E., Verga, P., Belanger, D., McCallum, A., "Fast and Accurate Entity Recognition with Iterated Dilated Convolutions," *EMNLP*, (2017).

[9] Chang, H.-S., Munir, A., Liu, A., Wei, J., Traylor, A., Nagesh, A., Monath, N., Verga, P., Strubell, E, McCallum, A., "Extracting Multilingual Relations under Limited Resources: TAC

2016 Cold-Start KB construction and Slot-Filling using Compositional Universal Schema," *Text Analysis Conference, Knowledge Base Population (TAC/KBP)*, (2016).

[10] Roth, B., Monath, N., Belanger, D., Strubell, E., Verga, P., McCallum, A., "Building Knowledge Bases with Universal Schema: Cold Start and Slot-Filling Approaches,", *Text Analysis Conference, Knowledge Base Population (TAC/KBP)*, (2015).

[11] Neelakantan, A., Shankar, J., Passos, A., McCallum, A., "Efficient Non-parametric Estimation of Multiple Embeddings per Word in Vector Space", *EMNLP*, (2014).

[12] Vilnis, L., McCallum, A. "Word Representations via Gaussian Embedding," ICLR, (2015).

[13] Neelakantan, A., Roth, B., McCallum, A., "Compositional Vector Space Models for Knowledge Base Completion," *ACL*, (2015).

[14] Das, R., Neelakantan, A., Belanger, D., McCallum, A., "Chains of Reasoning over Entities, Relations, and Text using Recurrent Neural Networks," *EACL*, (2017).

[15] Neelakantan, A., Le, Q., Sutskever, I.. "Neural programmer: Inducing latent programs with gradient descent," *ICLR* (2016).

[16] Neelakantan, A., Le, Q.V., Abadi, M., McCallum, A., "Learning a Natural Language Interface with Neural Programmer," *ICLR, (2017).*

[17] Suchanek, F., Kasneci, G. Weikum, G. "Yago: A core of semantic knowledge." *WWW*, (2007).

[18] Banko, M., Cafarella, M., Soderland, S., Broadhead, M., Etzioni, O. "Open information extraction from the web." IJCAI, (2007).

[19] Etzioni, O., Banko, M., Soderland, S, Weld, D. "Open information extraction from the web," *Communications of the ACM*, (2008).

[20] Yates,A. Etzioni, O. "Unsupervised resolution of objects and relations on the web." NAACL, (2007).

[21] Koren, Y. "Factorization meets the neighborhood: a multifaceted collaborative filtering model," *KDD*, (2008).

[22] Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-Thieme, L., "BPR: Bayesian personalized ranking from implicit feedback," UAI, (2009).

[23] Bahdanau, D., Cho, K., Bengio, Y., "Neural Machine Translation by Jointly Learning to Align and Translate." *ICLR*, (2015).

[24] Alex Graves, Greg Wayne, and Ivo Danihelka. "Neural Turing Machines." *arXiv preprint*

arxiv:1410.5401, (2014)

[25] Bunescu, R., Mooney, R., "Learning to extract relations from the web using minimal supervision," ACL, (2007).

[26] Mintz, M., Bills, S., Snow, R., Jurafsky, D. "Distant supervision for relation extraction without labeled data," IJCNLP, (2009).

[27] Riedel, S., Yao, L., McCallum, A. "Modeling relations and their mentions without labeled text," *Machine Learning and Knowledge Discovery in Databases*, (2010)

[28] Yao, L., Riedel, S., McCallum, A., "Collective cross-document relation extraction without labelled data," EMNLP, (2010).

[29] Hoffmann, R., Zhang, C., Ling, X. Zettlemoyer, L., Weld, D., "Knowledge-based weak supervision for information extraction of overlapping relations," *ACL*, (2011)

[30] Surdeanu, M., Tibshirani, J., Nallapati, R., Manning, C. " Multi-instance multi-label learning for relation extraction," *EMNLP* (2012)

[31] Min, B., Grishman, R., Wan, L., Wang, C. Gondek, D. "Distant supervision for relation extraction with an incomplete knowledge base," NAACL, (2013).

[32] Zeng, D., Liu, K., Chen, Y., Zhao, J. "Distant supervision for relation extraction via piecewise convolutional neural networks," *EMNLP*, (2015)

[33] Angeli, G., Gupta, S., Jose, M., Manning, C., Re, C., Tibshirani, J., Wu, J. Yu, S., Zhang, C. "Stanford's 2014 slot filling systems." TAC KBP, (2014)

[34] Yuan, M., Lin, Y., "Model selection and estimation in regression with grouped variables," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, (2006)

[35] Swirszcz, G., Abe, N., Lozano, A., "Grouped orthogonal matching pursuit for variable selection and prediction," *NIPS*, (2009)

[36] Lozano, A., Swirszcz, G., Abe, N., "Group orthogonal matching pursuit for logistic regression,"*AISTATS*, (2011)

[37] Kohavi, R., John, G., "Wrappers for feature subset selection," *Artificial Intelligence*, (1997)

[38] Mikolov, T., Chen, K., Corrado, G. Dean, J. "Efficient estimation of word representations in vector space." *CoRR, abs/1301.3781*, (2013).

[39] Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J. "Distributed representations of words and phrases and their compositionality." *NIPS*, (2013)

[40] Ratinov, L., Roth, D., “Design challenges and misconceptions in named entity recognition,” CoNLL, (2009).

[41] Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., Dyer, C., “Neural architectures for named entity recognition,” *NAACL*, (2016)

[42] Hochreiter, S. “The vanishing gradient problem during learning recurrent neural nets and problem solutions.” *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, (1998).

[43] Zheng, J., Vilnis, L, Singh, S., Choi, J., McCallum, A., “Dynamic Knowledge Base Alignment for Coreference Resolution,” CoNLL, (2013).

[44] Belanger, D., McCallum, A., “Structured Prediction Energy Networks”, *ICML*, (2016).

[45] Belanger, D., McCallum, A., “End-to-End Learning for Structured Prediction Energy Networks,” *ICML, (2017)*.

[46] Spitkovsky, V., Chang, A., “A cross-lingual dictionary for english wikipedia concepts,” *LREC*, (2012).

[47] Stoyanov, V. Eisner, J., “Easy-first coreference resolution.” COLING, (2012)

[48] Kobren, A., Monath, N., Krishnamurthy, A., McCallum, A., “A Hierarchical Algorithm for Extreme Clustering,” *KDD*, (2017).

[49] Heller, K., Ghahramani, Z. “Bayesian Hierarchical Clustering”, *ICML*, (2005).

[50] Aizerman, M. A., Braverman, E. A., and Rozonoer, L., “Theoretical foundations of the potential function method in pattern recognition learning.” *Automation and Remote Control*, 1964.

[51] Jebara, T., Kondor, R., and Howard, A., “Probability product kernels.” *JMLR*, (2004).

[52] Sandhaus, E. “The New York Times Annotated Corpus.” Linguistic Data Consortium, (2008).

[53] Toutanova, K., Chen, D., Pantel, P. Poon, H. Choudhury, “Representing text for joint embedding of text and knowledge bases,” *EMNLP*, (2015)

[54] Gabrilovich, E., Ringgaard, M., Subramanya, A., “Facc1: Freebase annotation of clueweb corpora, version 1” (release date 2013-06-26, format version 1, correction level 0). http://lemurproject.org/clueweb09/FACC1/, (2013)

[55] Ling, X., Weld, D. “Fine-grained entity recognition,” *AAAI*, (2012).

[56] Marcus, M., Santorini, B., Marcinkiewicz,M., “Building a Large Annotated Corpus of English: The Penn Treebank.” *Computational Linguistics,* 19(2):313–330. (1993)

[57] Choi, J. Palmer, M. “Getting the Most out of Transition-based Dependency Parsing,” ACL, *2011*.

[58] Lewis, D., Yang, Y., Rose, T., Li, F. “Rcv1: A new benchmark collection for text categorization research.” JMLR, (2004).

[59] Lin, D., Wu, X. “Phrase clustering for discriminative learning,” *ACL*, (2009)

[60] Durrett, G., Klein, D., “A joint model for entity analysis: Coreference, typing and linking,” TACL, (2014).

[61] Chiu, J., Nichols, E. “Named entity recognition with bidirectional LSTM-CNNs,” TACL, (2016).

[62] Doddington, G., Mitchell, A., Przybocki, M., Ramshaw, L., Strassel, S., Weischedel, R. “The Automatic Context Extraction (ACE) program-tasks,data, and evaluation,” LREC, (2004).

[63] Bengston,E., Roth, D., “Understanding the value of features for coreference resolution,” *EMNLP*, (2006).

[64] Dalton, J., Dietz, L., “A neighborhood relevance model for entity linking,” In Open Research Areas in Information Retrieval, (2013).

[65] Zhang, T., Ramakrishnan, R., Livny, M.. 1996. “BIRCH: An efficient data clustering method for very large databases.” SIGMOD, (1996).
D. Arthur and S. Vassilvitskii. 2007. k-means++: e advantages of careful seeding. In Symposium on Discrete Algorithms.

[66] Ackermann, M, Ma¨rtens, M., Raupach, C., Swierkot, K., Lammersen, C., Sohler. C., “StreamKM++: A clustering algorithm for data streams.” *Journal of Experimental Algorithmics*, (2012)

[67] Sculley, D. “Web-scale k-means clustering.” *WWW, (2010)*

[68] Fichtenberger, H. Gille´, M., Schmidt, M., Schwiegelshohn, C. Sohler, C., “BICO: BIRCH meets coresets for k-means clustering.” In European Symposium on Algorithms, (2013).

[69] Ester, M., Kriegel, H.-P., Sander, J., Xu, X. “A density-based algorithm for discovering clusters in large spatial databases with noise,” *KDD*, (1996).

[70] Geusebroek, J., Burghouts, G., Smeulders, A., “The Amsterdam library of object images,” *IJCV, (2005)*.

[71] Greenberg, C. S., Banse´, D., Doddington, G., Garcia-Romero, D., Godfrey, J., Kinnunen, T., Martin, A., McCree, A., Przybocki, M., Reynolds, D. "The NIST 2014 speaker recognition i-vector machine learning challenge." *Odyssey, (2014)*.

[72] Deng, J., Dong, W., Socher, R., Li, L., Li, K., Fei-Fei, L., "Imagenet: A large-scale hierarchical image database." *CVPR*, (2009).

[73] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M. "ImageNet Large Scale Visual Recognition Challenge." *IJCV*, (2015)

[74] Szegedy, C. ,Vanhoucke, V. , Iothe, S., Shlens, J., Wojna, Z., "Rethinking the inception architecture for computer vision," *CVPR*, (2016).

[75] Lichman. M. , UCI Machine Learning Repository. h p://archive.ics.uci.edu/ml (2013)

[76] Finkelstein, L., Gabrilovich, E., Matias, Y., Rivlin, E., Solan, Z., Wolfman,G., Ruppin, E., "Placing search in context: the concept revisited." *WWW* (2001).

[77] Huang, E., Socher, R., Manning, C., Ng, A. "Improving Word Representations via Global Context and Multiple Word Prototypes." *ACL*, (2012)

[78] Shaoul,C., Westbury,C., "The Westbury lab wikipedia corpus." (2010)

[79] Baroni, M., Bernardini, S., Ferraresi, A., Zanchetta, E. "The wacky wide web: a collection of very large linguistically processed web-crawled corpora." *Language resources and evaluation*, 43(3):209–226, (2009).

[80] Baroni, M., Bernardi, R., Do, N-Q, Shan, C-c. "Entailment above the word level in distributional semantics." *EACL*, (2012)

[81] Rubenstein, H., Goodenough, J. "Contextual correlates of synonymy." Commun. ACM, 8(10):627–633, October 1965. ISSN 0001-0782. (1965)

[82] Szumlanski, S. Gomez, F., Sims, V. "A new set of norms for semantic relatedness measures." *ACL*, 2013.

[83] Hill, F., Reichart, R., Korhonen, A. "Simlex-999: Evaluating semantic models with (genuine) similarity estimation." arXiv preprint arXiv:1408.3456, (2014).

[84] Miller, G, Charles, W. "Contextual correlates of semantic similarity." Language and cognitive processes, 6(1):1–28, (1991).

[85] Bruni, E., Tran, N.-K., and Baroni, M. "Multimodal distributional semantics." JAIR, (2014).

[86] Yang, D and Powers, D. "Verb similarity on the taxonomy of wordnet." *In the 3rd International WordNet Conference (GWC-06)*, Jeju Island, Korea, (2006).

[87] Faruqui, M and Dyer, C. “Community evaluation and exchange of word vectors at wordvectors.org.” *In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstration*s, Baltimore, USA, June 2014.

[88] Le, Q., Jaitly, N., Hinton, G. “A simple way to initialize recurrent networks of rectified linear units.” CoRR. (2015)

[89] Lao, N., Mitchell, T., Cohen, W. “Random walk inference and learning in a large scale knowledge base.” *EMNLP*, (2011).

[90] Pasupat, P., Liang, P., “Compositional semantic parsing on semi-structured tables.” ACL, (2015).

[91] Dai, A., Le, Q. “Semi-supervised sequence learning.” *NIPS*, (2015).

[92] Williams, R. “Simple statistical gradient-following algorithms for connectionist reinforcement learning.” *Machine Learning*, (1992).

APPENDIX A - Publications

2017 - Conference

Strubell, E., Verga, P., Belanger, D., McCallum, A., "Fast and Accurate Entity Recognition with Iterated Dilated Convolutions," *EMNLP*, (2017).

Kobren, A., Monath, N., Krishnamurthy, A., McCallum, A., "A Hierarchical Algorithm for Extreme Clustering," *KDD*, (2017).

Belanger, D., Yang, B., McCallum, A., "End-to-End Learning for Structured Prediction Energy Networks." *ICML*, (2017).

Das, R., Zaheer, M., Reddy, S., McCallum, A., "Question Answering on Knowledge Bases and Text using Universal Schema and Memory Networks," *ACL*, (2017).

Neelakantan, A., Le, Q.V., Abadi, M., McCallum, A., "Learning a Natural Language Interface with Neural Programmer," *ICLR, (2017).*

Das, R., Neelakantan, A., Belanger, D., McCallum, A., "Chains of Reasoning over Entities, Relations, and Text using Recurrent Neural Networks," *EACL*, (2017).

Verga, P., Neelakantan, A., McCallum, A., "Generalizing to Unseen Entities and Entity Pairs with Row-less Universal Schema," *EACL*, (2017).

2017 - Workshop

Strubell, E., McCallum, A. "Dependency Parsing with Dilated Iterated Graph CNNs", *2nd Workshop on Structured Prediction for Natural Language Processing (at EMNLP '17)*, (2017).

2016 - Conference

Verga, P., Belanger, D., Strubell, E., Roth, B., McCallum, A., "Multilingual Relation Extraction using Compositional Universal Schema," *NAACL*, (2016).

Belanger, D., McCallum, A., "Structured Prediction Energy Networks", *ICML*, (2016).

2016 - Workshop

Martin, T., Botschen,F., Nagesh, A., McCallum,A., "Call for Discussion: Building a New Standard Dataset for Relation Extraction Tasks." *Workshop on Automated Knowledge Base Construction (AKBC) at NAACL*, (2016).

Das, R., Neelakantan, A., Belanger, D., McCallum, A. "Incorporating Selectional Preferences in Multi-hop Relation Extraction," *Workshop on Automated Knowledge Base Construction (AKBC) at NAACL*, (2016).

Verga, P., McCallum, A., "Row-less Universal Schema", *Workshop on Automated Knowledge Base Construction (AKBC) at NAACL*, (2016).

Chang, H.-S., Munir, A., Liu, A., Wei, J., Traylor, A., Nagesh, A., Monath, N., Verga, P., Strubell, E, McCallum, A., "Extracting Multilingual Relations under Limited Resources: TAC 2016 Cold-Start KB construction and Slot-Filling using Compositional Universal Schema," *Text Analysis Conference, Knowledge Base Population (TAC/KBP)*, (2016).

2015 - Conference

Vilnis, L., McCallum, A. "Word Representations via Gaussian Embedding," ICLR, (2015).

Strubell, E. Vilnis, L., Silverstein, K., McCallum, A., "Learning Dynamic Feature Selection for Fast Sequential Prediction," *ACL*, (2015).

Neelakantan, A., Roth, B., McCallum, A., "Compositional Vector Space Models for Knowledge Base Completion," *ACL*, (2015).

2015 - Workshop

Roth, B., Monath, N., Belanger, D., Strubell, E., Verga, P., McCallum, A., "Building Knowledge Bases with Universal Schema: Cold Start and Slot-Filling Approaches,", *Text Analysis Conference, Knowledge Base Population (TAC/KBP)*, (2015).

2014 - Conference

Neelakantan, A., Shankar, J., Passos, A., McCallum, A., "Efficient Non-parametric Estimation of Multiple Embeddings per Word in Vector Space", *EMNLP*, (2014).

Passos, A., Kumar, V., McCallum, A., "Lexicon Infused Phrase Embeddings for Named Entity Resolution," *CoNLL*, (2014).

Belanger, D., Passos, A., Riedel, S., McCallum, A. "Message Passing for Soft Constraint Dual Decomposition," UAI, (2014).

2014 - Workshop

Strubell, E., Vilnis, L., McCallum, A. "Training for Fast Sequential Prediction Using Dynamic Feature Selection," *NIPS Workshop on Modern Machine Learning and NLP*, (2014).

Neelakantan, A., Roth, B., McCallum, A., "Knowledge Base Completion using Compositional Vector Space Models," *Workshop on Automated Knowledge Base Construction (AKBC)*, (2014).

Roth, B., Strubell, E., Silverstein, K., McCallum, A. "Minimally Supervised Event Argument Extraction using Universal Schema," *Workshop on Automated Knowledge Base Construction (AKBC) at NIPS*, (2014).

Roth, B., Strubell, E., Sullivan, J., Vikraman, L., Silverstein, K., McCallum, A. "Universal Schema for Slot-Filling, Cold-Start KBP and Event Argument Extraction: UMass IESL at TAC KBP 2014." *Text Analysis Conference (Knowledge Base Population Track)* (TAC KBP), (2014).

Neelakantan, A., Passos, A., McCallum, A., "A Hierarchical Model for Universal Schema Relation Extraction," *Workshop on Automatic Creation and Curation of Knowledge Bases (WACCK) at SIGMOD*, (2014).

2013 - Conference

Riedel, S., Yao, L., Marlin, B., McCallum, A. "Relation Extraction with Matrix Factorization and Universal Schemas." *NAACL*, (2013).

Choi, J., McCallum, A., "Transition-based Dependency Parsing with Selectional Branching,", ACL, (2013).

Zheng, J., Vilnis, L, Singh, S., Choi, J., McCallum, A., "Dynamic Knowledge Base Alignment for Coreference Resolution," *CoNLL*, (2013).

2013 - Workshop

Passos, A., Vilnis, L., McCallum, A., "Optimization and Learning in FACTORIE," *Neural Information Processing Systems Workshop on Optimization for Machine Learning*, (2013)
Belanger, D., Sheldon, D., McCallum, A., "Marginal Inference in MRFs using Frank-Wolfe," *Neural Information Processing Systems Workshop on Greedy Optimization, Frank-Wolfe and Friends*, (2013).

Singh, S., Belanger, D., Kobren, A., Wick, M., Passos, A., Pandya, H., Choi, J., Martin, B., McCallum, A., "Universal Schema for Slot Filling and Cold Start: UMass IESL at TACKBP." *Text Analysis Conference (TAC)*, (2013).

Yao, L., Riedel, S., Marlin, B., McCallum, A., "Universal Schema for Entity Type Prediction," *Workshop on Automated Knowledge Base Construction (AKBC)*, (2013).

Wick, M., Singh, S., Pandya, H., McCallum, A., "A Joint Model for Discovering and Linking Entities," *Workshop on Automated Knowledge Base Construction (AKBC)*, (2013).

Singh, S., Riedel, S., Martin, B., Zheng, J., McCallum, A., "Joint Inference of Entities, Relations, and Coreference," *Workshop on Automated Knowledge Base Construction (AKBC)*, (2013).

APPENDIX B - Abstracts of Conference Papers

2017

Strubell, E., Verga, P., Belanger, D., McCallum, A., "Fast and Accurate Entity Recognition with Iterated Dilated Convolutions," *EMNLP*, (2017).

Today when many practitioners run basic NLP on the entire web and large-volume traffic, faster methods are paramount to saving time and energy costs. Recent advances in GPU hardware have led to the emergence of bi-directional LSTMs as a standard method for obtaining per-token vector representations serving as input to labeling tasks such as NER (often followed by prediction in a linear-chain CRF). Though expressive and accurate, these models fail to fully exploit GPU parallelism, limiting their computational efficiency. This paper proposes a faster alternative to Bi-LSTMs for NER: Iterated Dilated Convolutional Neural Networks (ID-CNNs), which have better capacity than traditional CNNs for large context and structured prediction. Unlike LSTMs whose sequential processing on sentences of length N requires O(N) time even in the face of parallelism, ID-CNNs permit fixed-depth convolutions to run in parallel across entire documents. We describe a distinct combination of network structure, parameter sharing and training procedures that enable dramatic 14-20x test-time speedups while retaining accuracy comparable to the Bi-LSTM-CRF. Moreover, ID-CNNs trained to aggregate context from the entire document are even more accurate while maintaining 8x faster test time speeds.

Kobren, A., Monath, N., Krishnamurthy, A., McCallum, A., "A Hierarchical Algorithm for Extreme Clustering," *KDD*, (2017).

Many modern clustering methods scale well to a large number of data points, N, but not to a large number of clusters, K. This paper introduces PERCH, a new non-greedy, incremental algorithm for hierarchical clustering that scales to both massive N and K---a problem setting we term extreme clustering. Our algorithm efficiently routes new data points to the leaves of an incrementally-built tree. Motivated by the desire for both accuracy and speed, our approach performs tree rotations for the sake of enhancing subtree purity and encouraging balancedness. We prove that, under a natural separability assumption, our non-greedy algorithm will produce trees with perfect dendrogram purity regardless of data arrival order. Our experiments demonstrate that PERCH constructs more accurate trees than other tree-building clustering algorithms and scales well with both N and K, achieving a higher quality clustering than the strongest flat clustering competitor in nearly half the time.

Belanger, D., Yang, B., McCallum, A., "End-to-End Learning for Structured Prediction Energy Networks." *ICML*, (2017).

Structured Prediction Energy Networks (SPENs) are a simple, yet expressive family of structured prediction models (Belanger and McCallum, 2016). An energy function over candidate structured outputs is given by a deep network, and predictions are formed by gradient-based optimization. This paper presents end-to-end learning for SPENs, where the energy function is discriminatively

trained by back-propagating through gradient-based prediction. In our experience, the approach is substantially more accurate than the structured SVM method of Belanger and McCallum (2016), as it allows us to use more sophisticated non-convex energies. We provide a collection of techniques for improving the speed, accuracy, and memory requirements of end-to-end SPENs, and demonstrate the power of our method on 7-Scenes image denoising and CoNLL-2005 semantic role labeling tasks. In both, inexact minimization of non-convex SPEN energies is superior to baseline methods that use simplistic energy functions that can be minimized exactly.

Das, R., Zaheer, M., Reddy, S., McCallum, A., "Question Answering on Knowledge Bases and Text using Universal Schema and Memory Networks," *ACL,* (2017).

Existing question answering methods infer answers either from a knowledge base or from raw text. While knowledge base (KB) methods are good at answering compositional questions, their performance is often affected by the incompleteness of the KB. Au contraire, web text contains millions of facts that are absent in the KB, however in an unstructured form. Universal schema can support reasoning on the union of both structured KBs and unstructured text by aligning them in a common embedded space. In this paper we extend universal schema to natural language question answering, employing memory networks to attend to the large body of facts in the combination of text and KB. Our models can be trained in an end-to-end fashion on question-answer pairs. Evaluation results on SPADES fill-in-the-blank question answering dataset show that exploiting universal schema for question answering is better than using either a KB or text alone. This model also outperforms the current state-of-the-art by 8.5 F1 points.

Neelakantan, A., Le, Q.V., Abadi, M., McCallum, A., "Learning a Natural Language Interface with Neural Programmer," *ICLR, (2017).*

Learning a natural language interface for database tables is a challenging task that involves deep language understanding and multi-step reasoning. The task is often approached by mapping natural language queries to logical forms or programs that provide the desired response when executed on the database. To our knowledge, this paper presents the first weakly supervised, end-to-end neural network model to induce such programs on a real-world dataset. We enhance the objective function of Neural Programmer, a neural network with built-in discrete operations, and apply it on WikiTableQuestions, a natural language question-answering dataset. The model is trained end-to-end with weak supervision of question-answer pairs, and does not require domain-specific grammars, rules, or annotations that are key elements in previous approaches to program induction. The main experimental result in this paper is that a single Neural Programmer model achieves 34.2% accuracy using only 10,000 examples with weak supervision. An ensemble of 15 models, with a trivial combination technique, achieves 37.7% accuracy, which is competitive to the current state-of-the-art accuracy of 37.1% obtained by a traditional natural language semantic parser.

Das, R., Neelakantan, A., Belanger, D., McCallum, A., "Chains of Reasoning over Entities, Relations, and Text using Recurrent Neural Networks," *EACL,* (2017).

Our goal is to combine the rich multistep inference of symbolic logical reasoning with the generalization capabilities of neural networks. We are particularly interested in complex reasoning about entities and relations in text and large-scale knowledge bases (KBs). Neelakantan et al. (2015) use RNNs to compose the distributed semantics of multi-hop paths in KBs; however for multiple reasons, the approach lacks accuracy and practicality. This paper proposes three significant modeling advances: (1) we learn to jointly reason about relations, entities, and entity-types; (2) we use neural attention modeling to incorporate multiple paths; (3) we learn to share strength in a single RNN that represents logical composition across all relations. On a large scale Freebase+ClueWeb prediction task, we achieve 25% error reduction, and a 53% error reduction on sparse relations due to shared strength. On chains of reasoning in WordNet we reduce error in mean quantile by 84% versus previous state-of-the-art.

Verga, P., Neelakantan, A., McCallum, A., "Generalizing to Unseen Entities and Entity Pairs with Row-less Universal Schema," ***EACL*****, (2017).**

Universal schema predicts the types of entities and relations in a knowledge base (KB) by jointly embedding the union of all available schema types—not only types from multiple structured databases (such as Freebase or Wikipedia infoboxes), but also types expressed as textual patterns from raw text. This prediction is typically modeled as a matrix completion problem, with one type per column, and either one or two entities per row (in the case of entity types or binary relation types, respectively). Factorizing this sparsely observed matrix yields a learned vector embedding for each row and each column. In this paper we explore the problem of making predictions for entities or entity-pairs unseen at training time (and hence without a pre-learned row embedding). We propose an approach having no per-row parameters at all; rather we produce a row vector on the fly using a learned aggregation function of the vectors of the observed columns for that row. We experiment with various aggregation functions, including neural network attention models. Our approach can be understood as a natural language database, in that questions about KB entities are answered by attending to textual or database evidence. In experiments predicting both relations and entity types, we demonstrate that despite having an order of magnitude fewer parameters than traditional universal schema, we can match the accuracy of the traditional model, and more importantly, we can now make predictions about unseen rows with nearly the same accuracy as rows available at training time.

2016 - Conference

Verga, P., Belanger, D., Strubell, E., Roth, B., McCallum, A., "Multilingual Relation Extraction using Compositional Universal Schema," ***NAACL*****, (2016).**

Universal schema builds a knowledge base (KB) of entities and relations by jointly embedding all relation types from input KBs as well as textual patterns observed in raw text. In most previous applications of universal schema, each textual pattern is represented as a single embedding, preventing generalization to unseen patterns. Recent work employs a neural network to capture patterns' compositional semantics, providing generalization to all possible input text. In response, this paper introduces significant further improvements to the coverage and flexibility of universal schema relation extraction: predictions for entities unseen in training and multilingual transfer learning to domains with no annotation. We evaluate our model through extensive experiments on

the English and Spanish TAC KBP benchmark, outperforming the top system from TAC 2013 slot-filling using no handwritten patterns or additional annotation. We also consider a multilingual setting in which English training data entities overlap with the seed KB, but Spanish text does not. Despite having no annotation for Spanish data, we train an accurate predictor, with additional improvements obtained by tying word embeddings across languages. Furthermore, we find that multilingual training improves English relation extraction accuracy. Our approach is thus suited to broad-coverage automated knowledge base construction in a variety of languages and domains.

Belanger, D., McCallum, A., "Structured Prediction Energy Networks", *ICML*, (2016).

We introduce structured prediction energy networks (SPENs), a flexible framework for structured prediction. A deep architecture is used to define an energy function of candidate labels, and then predictions are produced by using back-propagation to iteratively optimize the energy with respect to the labels. This deep architecture captures dependencies between labels that would lead to intractable graphical models, and performs structure learning by automatically learning discriminative features of the structured output. One natural application of our technique is multi-label classification, which traditionally has required strict prior assumptions about the interactions between labels to ensure tractable learning and prediction. We are able to apply SPENs to multi-label problems with substantially larger label sets than previous applications of structured prediction, while modeling high-order interactions using minimal structural assumptions. Overall, deep learning provides remarkable tools for learning features of the inputs to a prediction problem, and this work extends these techniques to learning features of structured outputs. Our experiments provide impressive performance on a variety of benchmark multi-label classification tasks, demonstrate that our technique can be used to provide interpretable structure learning, and illuminate fundamental trade-offs between feed-forward and iterative structured prediction.

2015

Vilnis, L., McCallum, A. "Word Representations via Gaussian Embedding," *ICLR*, (2015).

Current work in lexical distributed representations maps each word to a point vector in low-dimensional space. Mapping instead to a density provides many interesting advantages, including better capturing uncertainty about a representation and its relationships, expressing asymmetries more naturally than dot product or cosine similarity, and enabling more expressive parameterization of decision boundaries. This paper advocates for density-based distributed embeddings and presents a method for learning representations in the space of Gaussian distributions. We compare performance on various word embedding benchmarks, investigate the ability of these embeddings to model entailment and other asymmetric relationships, and explore novel properties of the representation.

Strubell, E. Vilnis, L., Silverstein, K., McCallum, A., "Learning Dynamic Feature Selection for Fast Sequential Prediction," *ACL*, (2015).

We present paired learning and inference algorithms for significantly reducing computation and increasing speed of the vector dot products in the classifiers that are at the heart of many NLP components. This is accomplished by partitioning the features into a sequence of templates which are ordered such that high confidence can often be reached using only a small fraction of all features. Parameter estimation is arranged to maximize accuracy and early confidence in this sequence. Our approach is simpler and better suited to NLP than other related cascade methods. We present experiments in left-to-right part-of-speech tagging, named entity recognition, and transition-based dependency parsing. On the typical benchmarking datasets we can preserve POS tagging accuracy above 97% and parsing LAS above 88.5% both with over a five-fold reduction in run-time, and NER F1 above 88 with more than 2x increase in speed.

Neelakantan, A., Roth, B., McCallum, A., "Compositional Vector Space Models for Knowledge Base Completion," ***ACL*****, (2015).**

Knowledge base (KB) completion adds new facts to a KB by making inferences from existing facts, for example by inferring with high likelihood nationality(X,Y) from bornIn(X,Y). Most previous methods infer simple one-hop relational synonyms like this, or use as evidence a multi-hop relational path treated as an atomic feature, like bornIn(X,Z) -> containedIn(Z,Y). This paper presents an approach that reasons about conjunctions of multi-hop relations non-atomically, composing the implications of a path using a recursive neural network (RNN) that takes as inputs vector embeddings of the binary relation in the path. Not only does this allow us to generalize to paths unseen at training time, but also, with a single high-capacity RNN, to predict new relation types not seen when the compositional model was trained (zero-shot learning). We assemble a new dataset of over 52M relational triples, and show that our method improves over a traditional classifier by 11%, and a method leveraging pre-trained embeddings by 7%.

2014

Neelakantan, A., Shankar, J., Passos, A., McCallum, A., "Efficient Non-parametric Estimation of Multiple Embeddings per Word in Vector Space", ***EMNLP*****, (2014).**

There is rising interest in vector-space word embeddings and their use in NLP, especially given recent methods for their fast estimation at very large scale. Nearly all this work, however, assumes a single vector per word type—ignoring polysemy and thus jeopardizing their usefulness for downstream tasks. We present an extension to the Skip-gram model that efficiently learns multiple embeddings per word type. It differs from recent related work by jointly performing word sense discrimination and embedding learning, by non-parametrically estimating the number of senses per word type, and by its efficiency and scalability. We present new state-of-the-art results in the word similarity in context task and demonstrate its scalability by training with one machine on a corpus of nearly 1 billion tokens in less than 6 hours.

Passos, A., Kumar, V., McCallum, A., "Lexicon Infused Phrase Embeddings for Named Entity Resolution," ***CoNLL*****, (2014).**

Most state-of-the-art approaches for named-entity recognition (NER) use semi supervised information in the form of word clusters and lexicons. Recently neural network-based language models have been explored, as they as a byproduct generate highly informative vector representations for words, known as word embeddings. In this paper we present two contributions: a new form of learning word embeddings that can leverage information from relevant lexicons to improve the representations, and the first system to use neural word embeddings to achieve state-of-the-art results on named-entity recognition in both CoNLL and Ontonotes NER. Our system achieves an F1 score of 90.90 on the test set for CoNLL 2003—significantly better than any previous system trained on public data, and matching a system employing massive private industrial query-log data.

Belanger, D., Passos, A., Riedel, S., McCallum, A. "Message Passing for Soft Constraint Dual Decomposition," UAI, (2014).

Dual decomposition provides the opportunity to build complex, yet tractable, structured prediction models using linear constraints to link together submodels that have available MAP inference routines. However, since some constraints might not hold on every single example, such models can often be improved by relaxing the requirement that these constraints always hold, and instead replacing them with soft constraints that merely impose a penalty if violated. A dual objective for the resulting MAP inference problem differs from the hard constraint problem's associated dual decomposition objective only in that the dual variables are subject to box constraints. This paper introduces a novel primal dual block coordinate descent algorithm for minimizing this general family of box-constrained objectives. Through experiments on two natural language corpus-wide inference tasks, we demonstrate the advantages of our approach over the current alternative, based on copying variables, adding auxiliary submodels and using traditional dual decomposition. Our algorithm performs inference in the same model as was previously published for these tasks, and thus is capable of achieving the same accuracy, but provides a 2-10x speedup over the current state of the art.

2013

Riedel, S., Yao, L., Marlin, B., McCallum, A. "Relation Extraction with Matrix Factorization and Universal Schemas." ***NAACL*****, (2013).**

Traditional relation extraction predicts relations within some fixed and finite target schema. Machine learning approaches to this task require either manual annotation or, in the case of distant supervision, existing structured sources of the same schema. The need for existing datasets can be avoided by using a universal schema: the union of all involved schemas (surface form predicates as in OpenIE, and relations in the schemas of preexisting databases). This schema has an almost unlimited set of relations (due to surface forms), and supports integration with existing structured data (through the relation types of existing databases). To populate a database of such schema we present matrix factorization models that learn latent feature vectors for entity tuples and relations. We show that such latent models achieve substantially higher accuracy than a traditional classification approach. More importantly, by operating simultaneously on relations observed in text and in preexisting structured DBs such as Freebase, we are able to reason about

unstructured and structured data in mutually-supporting ways. By doing so our approach outperforms state-of-the-art distant supervision.

Choi, J., McCallum, A., "Transition-based Dependency Parsing with Selectional Branching," ACL, (2013).

We present a novel approach, called selectional branching, which uses confidence estimates to decide when to employ a beam, providing the accuracy of beam search at speeds close to a greedy transition-based dependency parsing approach. Selectional branching is guaranteed to perform a fewer number of transitions than beam search yet performs as accurately. We also present a new transition-based dependency parsing algorithm that gives a complexity of O(n) for projective parsing and an expected linear time speed for non-projective parsing. With the standard setup, our parser shows an unlabeled attachment score of 92.96% and a parsing speed of 9 milliseconds per sentence, which is faster and more accurate than the current state-of-the-art transition based parser that uses beam search.

Zheng, J., Vilnis, L, Singh, S., Choi, J., McCallum, A., "Dynamic Knowledge Base Alignment for Coreference Resolution," *CoNLL*, (2013).

Coreference resolution systems can benefit greatly from inclusion of global context, and a number of recent approaches have demonstrated improvements when precomputing an alignment to external knowledge sources. However, since alignment itself is a challenging task and is often noisy, existing systems either align conservatively, resulting in very few links, or combine the attributes of multiple candidates, leading to a conflation of entities. Our approach instead performs joint inference between within-document coreference and entity linking, maintaining ranked lists of candidate entities that are dynamically merged and reranked during inference. Further, we incorporate a large set of surface string variations for each entity by using anchor texts from the web that link to the entity. These forms of global context enables our system to improve classifier-based coreference by 1.09 B^3 F1 points, and improve over the previous state-of-art by 0.41 points, thus introducing a new state-of-art result on the ACE 2004 data.

LIST OF SYMBOLS, ABBREVIATIONS, AND ACRONYMS

AKBC	Automatic Knowledge Base Construction
KB	Knowledge Base
NER	Named Entity Recognition
CRF	Conditional Random Field
LAS	Label Attachment Score
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
Bi-LSTM-CRF	Bi-directional LSTM Conditional Random Field
CNN	Convolutional Neural Network
ID-CNN	Iterated Dilated Convolutional Neural Network
SPEN	Structured Prediction Energy Network
PERCH	Purity Enhancing Rotations for Cluster Hierarchies
LCA	Least Common Ancestor
KL-Divergence	Kullback-Leibler Divergence
SF	Slot Filling
MRR	Mean Reciprocal Rank